

PATENT COOPERATION TREATY

PCT

NOTIFICATION CONCERNING
AMENDMENTS OF THE CLAIMS(PCT Rule 62 and
Administrative Instructions, Section 417)

From the INTERNATIONAL BUREAU

To:

Commissioner
US Department of Commerce
United States Patent and Trademark
Office, PCT
2011 South Clark Place Room
CP2/5C24
Arlington, VA 22202
ETATS-UNIS D'AMERIQUE
in its capacity as International Preliminary Examining Authority

Date of mailing (day/month/year)

12 December 2000 (12.12.00)

International application No.

PCT/IL00/00178

International filing date (day/month/year)

20 March 2000 (20.03.00)

Applicant

INTERLINK COMPUTER COMMUNICATIONS LTD. et al

The International Bureau hereby informs the International Preliminary Examining Authority that no amendments under Article 19 have been received by the International Bureau (Administrative Instructions, Section 417).

The International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland

Facsimile No. (41-22) 740.14.35

Authorized officer

F. Baechler

Telephone No. (41-22) 338.83.38

PCT COOPERATION TREATY

PCT

NOTIFICATION OF ELECTION

(PCT Rule 61.2)

From the INTERNATIONAL BUREAU

To:

Commissioner
 US Department of Commerce
 United States Patent and Trademark
 Office, PCT
 2011 South Clark Place Room
 CP2/5C24
 Arlington, VA 22202
 ETATS-UNIS D'AMERIQUE
 in its capacity as elected Office

Date of mailing (day/month/year) 12 December 2000 (12.12.00)	
International application No. PCT/IL00/00178	Applicant's or agent's file reference I01/1
International filing date (day/month/year) 20 March 2000 (20.03.00)	Priority date (day/month/year) 22 March 1999 (22.03.99)
Applicant HADAS, Ilan	

1. The designated Office is hereby notified of its election made:

☒ in the demand filed with the International Preliminary Examining Authority on:
 20 October 2000 (20.10.00)

☐ in a notice effecting later election filed with the International Bureau on:

2. The election ☒ was
☐ was not

made before the expiration of 19 months from the priority date or, where Rule 32 applies, within the time limit under Rule 32.2(b).

The International Bureau of WIPO 34, chemin des Colombettes 1211 Geneva 20, Switzerland Facsimile No.: (41-22) 740.14.35	Authorized officer F. Baechler Telephone No.: (41-22) 338.83.38
---	---

WHAT IS CLAIMED IS:

1. A system for automatic construction of an application for an enterprise resource for operation by a user, the system comprising:
 - (a) a meta-data description of the enterprise resource, said meta-data description featuring a plurality of logical units of data;
 - (b) a meta-data parser for parsing the enterprise resource into said plurality of logical units of data according to said meta-data description and for determining a relationship between said logical units of data;
 - (c) a meta-data translator for translating each logical unit of data into an application component, for associating said application component with at least one interface component for interfacing with the enterprise resource; and
 - (d) a definition extraction factory for creating the application from a plurality of said application components with said at least one interface component, at least according to said relationship between said plurality of logical units of data.
2. The system of claim 1, wherein said at least one interface component is a plurality of interface components, including at least one interface component for data flow to and from the enterprise resource and at least one interface component for providing a user interface to the user.
3. The system of claim 2, wherein the application further comprises:
 - (i) a first layer of interface components for providing control of the application, including at least one interface component for controlling said user interface and at least one interface component for controlling said data flow;
 - (ii) a second layer of interface components for providing said data input and said data output, including said at least one interface component for providing said user interface; and
 - (iii) a third layer for specifically interacting with the enterprise resource.
4. The system of claim 3, wherein said application component is contained in said third layer and said third layer further comprises:
 - (1) an export interface for retrieving data from the enterprise resource; and
 - (2) an import interface for storing data into the enterprise resource.

5. The system of claim 4, wherein each of said first layer and said second layer includes an extension of said application component for communicating with said at least one interface component related to said user interface and said at least one interface component related to said data flow.

6. The system of claim 5, wherein said at least one interface component is defined according to a standard interface and according to at least one user-defined function.

7. The system of claim 1, wherein said meta-data description, said meta-data parser and said meta-data translator are specific for a particular type of enterprise resource.

8. The system of claim 7, wherein said meta-data translator translates each logical unit of data according to at least one internal translation rule.

9. The system of claim 8, wherein said at least one internal translation rule is defined according to an attribute of said particular type of enterprise resource.

10. The system of claim 9, wherein said at least one internal translation rule further includes at least one default value for a parameter not described in said meta-data description.

11. The system of claim 10, wherein said meta-data translator also translates each logical unit of data according to at least one external translation rule defined by the user.

12. The system of claim 1, further comprising:

- (e) a hierarchical structure constructor for arranging said plurality of application components into a hierarchical object-oriented structure according to said relationship between said logical units of data, such that said definition extraction factory also creates the application according to said hierarchical object-oriented structure.

13. The system of claim 12, further comprising:

- (f) a definitions file for defining each of said plurality of application components, such that the application is stored as said definitions file, said definitions file being constructed by said definition extraction factory.
- 14. The system of claim 13, further comprising:
 - (g) an editing environment for editing said hierarchical object-oriented structure by the user, such that the user manually alters at least a portion of said hierarchical object oriented structure.
- 15. The system of claim 14, further comprising:
 - (h) a parsing engine for reading said definitions file and for operating the application according to said definitions file.
- 16. The system of claim 15, further comprising:
 - (i) a user interface of the application for interacting with the user; and
 - (j) a client for operating said user interface, such that said parsing engine constructs said user interface automatically according to at least one property of said client.
- 17. The system of claim 16, wherein said user interface is a GUI (graphical user interface), said GUI being constructed by said meta-data translator with at least one abstract GUI component, the system further comprising:
 - (i) at least one GUI library for containing a conversion of said at least one abstract GUI component to a concrete GUI component, such that said parsing engine also constructs said user interface automatically according to said GUI library.
- 18. The system of claim 17, wherein said at least one GUI library contains a plurality of concrete GUI components and a layout manager for determining a spatial relationship between said plurality of concrete GUI components.
- 19. A method for automatically constructing an application for an enterprise data resource for a user, the steps of the method being performed by a data processor, the method comprising the steps of:

- (a) providing a meta-data description of the enterprise resource, said meta-data description including at least one attribute of the enterprise resource;
 - (b) dividing the enterprise resource into a plurality of logical data units according to said meta-data description;
 - (c) translating each of said plurality of logical data units into an application component according to at least one attribute of said meta-data description to form a plurality of application components;
 - (d) determining a relationship between said plurality of logical data units; and
 - (e) constructing the application according to said plurality of application components and said relationship between said plurality of logical data units.
20. The method of claim 19, wherein step (d) further comprises the step of:
- (i) constructing a hierarchical object-oriented structure for said plurality of application components according to said relationship between said plurality of logical data units.
21. The method of claim 20, wherein step (d) further comprises the steps of:
- (ii) providing an editing environment for displaying said hierarchical object-oriented structure to the user; and
 - (iii) altering at least a portion of said hierarchical object-oriented structure by the user.
22. The method of claim 21, wherein step (c) is performed according to at least one internal translation rule, said at least one internal translation rule being defined according to a type of the enterprise resource.
23. The method of claim 22, wherein step (c) is additionally performed according to the steps of:
- (i) defining at least one external translation rule by the user; and
 - (ii) translating each of said plurality of logical data units into an application component according to said at least one external translation rule.

24. The method of claim 23, further comprising the steps of:
- (f) extracting a plurality of definitions for describing each of said plurality of application components and said relationship; and
 - (g) storing the application by storing said plurality of definitions.
25. The method of claim 24, further comprising the step of:
- (h) operating the application according to said plurality of definitions.
26. The method of claim 25, further comprising the step of:
- (i) altering the application by changing at least one of said plurality of definitions.
27. The method of claim 22, wherein the application is operated by a client, and step (e) includes the steps of:
- (1) automatically constructing a user interface according to at least one property of said client; and
 - (2) displaying said user interface to the user for interacting with the user.
28. The method of claim 27, wherein step (1) further includes the steps of:
- (A) constructing an abstract user interface for displaying data from the enterprise resource and for interacting with the user; and
 - (B) constructing a concrete user interface according to said abstract user interface and according to said at least one property of said client.

PATENT COOPERATION TREATY

PCT

RECD 04 MAY 2001

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

PCT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference I01/1	FOR FURTHER ACTION See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)	
International application No. PCT/IL00/00178	International filing date (day/month/year) 20 MARCH 2000	Priority date (day/month/year) 22 MARCH 1999
International Patent Classification (IPC) or national classification and IPC IPC(7): G06F 17/30 and US Cl.: 707/2		
Applicant INTERLINK COMPUTER COMMUNICATIONS LTD.		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.
2. This REPORT consists of a total of 4 sheets.
- ☐ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority. (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

These annexes consist of a total of 0 sheets.

3. This report contains indications relating to the following items:

- I ☒ Basis of the report
- II ☐ Priority
- III ☐ Non-establishment of report with regard to novelty, inventive step or industrial applicability
- IV ☐ Lack of unity of invention
- V ☒ Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- VI ☐ Certain documents cited
- VII ☐ Certain defects in the international application
- VIII ☐ Certain observations on the international application

Date of submission of the demand 20 OCTOBER 2000	Date of completion of this report 20 APRIL 2001
Name and mailing address of the IPEA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer MARK POWELL <i>Peggy Hanood</i>
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9703

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.

PCT/IL00/00178

I. Basis of the report

1. With regard to the elements of the international application: *

☒ the international application as originally filed☒ the description:

pages 1-15 , as originally filed
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

☒ the claims:

pages 16-20 , as originally filed
pages NONE , as amended (together with any statement) under Article 19
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

☒ the drawings:

pages 1-2 , as originally filed
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

☒ the sequence listing part of the description:

pages NONE , as originally filed
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

2. With regard to the language, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.

These elements were available or furnished to this Authority in the following language _____ which is:

- ☐ the language of a translation furnished for the purposes of international search (under Rule 23.1(b)).
☐ the language of publication of the international application (under Rule 48.3(b)).
☐ the language of the translation furnished for the purposes of international preliminary examination (under Rules 55.2 and/or 55.3).

3. With regard to any nucleotide and/or amino acid sequence disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:

- ☐ contained in the international application in printed form.
☐ filed together with the international application in computer readable form.
☐ furnished subsequently to this Authority in written form.
☐ furnished subsequently to this Authority in computer readable form.
☐ The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
☐ The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.

4. ☒ The amendments have resulted in the cancellation of:

☒ the description, pages NONE
☒ the claims, Nos. NONE
☒ the drawings, sheets/~~fig~~ NONE

5. ☐ This report has been drawn as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).**

* Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17).

**Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.

PCT/IL00/00178

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**1. statement**

Novelty (N)	Claims	<u>NONE</u>	YES
	Claims	<u>1-28</u>	NO
Inventive Step (IS)	Claims	<u>NONE</u>	YES
	Claims	<u>1-28</u>	NO
Industrial Applicability (IA)	Claims	<u>NONE</u>	YES
	Claims	<u>1-28</u>	NO

2. citations and explanations (Rule 70.7)

Claims 1-28 lack novelty under PCT Article 33(2) as being anticipated by Coyler's "Bridge for a Client-Server Environment."

The claims are directed to a standard tool-of-the-trade known generally as a "wrapper." The specific implementation claimed is known more specifically as an "Object Request Broker" (ORB). Coyler's Figure 1 shows the use of ORBs in a client-server environment, as claimed by Applicant.

Claims 1-28 lack an inventive step under PCT Article 33(3) as being obvious over an ORB, as taught by Coyler. Applicant's claims relate to a standard tool-of-the-trade in database application development, as taught by Coyler.

Claims 1-28 lack industrial applicability as defined by PCT Article 33(4). The claimed method is not limited to a practical application. As such, the claimed invention is clearly ineligible for patent protection.

----- NEW CITATIONS -----
NONE

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.

PCT/IL00/00178

Supplemental Box

(To be used when the space in any of the preceding boxes is not sufficient)

Continuation of: Boxes I - VIII

Sheet 10

PATENT COOPERATION TREATY

From the
INTERNATIONAL PRELIMINARY EXAMINING AUTHORITY

To: D'VORAH GRAESER
C/O THE POLKINGHORNS
9003 FLORIN WAY
UPPER MARLBORO, MD 20772

PCT

NOTIFICATION OF TRANSMITTAL OF INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Rule 71.1)

Date of Mailing
(day/month/year) **17 JUL 2001**

Applicant's or agent's file reference
I01/1

IMPORTANT NOTIFICATION

International application No.
PCT/IL00/00178

International filing date (day/month/year)
20 MARCH 2000

Priority Date (day/month/year)
22 MARCH 1999

Applicant
INTERLINK COMPUTER COMMUNICATIONS LTD.

1. The applicant is hereby notified that this International Preliminary Examining Authority transmits herewith the international preliminary examination report and its annexes, if any, established on the international application.
2. A copy of the report and its annexes, if any, is being transmitted to the International Bureau for communication to all the elected Offices.
3. Where required by any of the elected Offices, the International Bureau will prepare an English translation of the report (but not of any annexes) and will transmit such translation to those Offices.
4. **REMINDER**

The applicant must enter the national phase before each elected Office by performing certain acts (filing translations and paying national fees) within 30 months from the priority date (or later in some Offices)(Article 39(1))(see also the reminder sent by the International Bureau with Form PCT/IB/301).

Where a translation of the international application must be furnished to an elected Office, that translation must contain a translation of any annexes to the international preliminary examination report. It is the applicant's responsibility to prepare and furnish such translation directly to each elected Office concerned.

For further details on the applicable time limits and requirements of the elected Offices, see Volume II of the PCT Applicant's Guide.

Name and mailing address of the IPEA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

GEORGE DAVIS

Peggy Hance

Telephone No. (703) 305-3891

PATENT COOPERATION TREATY

From the
INTERNATIONAL PRELIMINARY EXAMINING AUTHORITY

To: D'VORAH GRAESER
C/O THE POLKINGHORNS
9003 FLORIN WAY
UPPER MARLBORO, MD 20772

PCT

NOTIFICATION OF TRANSMITTAL OF INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Rule 71.1)

Date of Mailing
(day/month/year) **17 JUL 2001**

Applicant's or agent's file reference
I01/1

IMPORTANT NOTIFICATION

International application No.
PCT/IL00/00178

International filing date (day/month/year)
20 MARCH 2000

Priority Date (day/month/year)
22 MARCH 1999

Applicant
INTERLINK COMPUTER COMMUNICATIONS LTD.

1. The applicant is hereby notified that this International Preliminary Examining Authority transmits herewith the international preliminary examination report and its annexes, if any, established on the international application.
2. A copy of the report and its annexes, if any, is being transmitted to the International Bureau for communication to all the elected Offices.
3. Where required by any of the elected Offices, the International Bureau will prepare an English translation of the report (but not of any annexes) and will transmit such translation to those Offices.
4. **REMINDER**

The applicant must enter the national phase before each elected Office by performing certain acts (filing translations and paying national fees) within 30 months from the priority date (or later in some Offices)(Article 39(1))(see also the reminder sent by the International Bureau with Form PCT/IB/301).

Where a translation of the international application must be furnished to an elected Office, that translation must contain a translation of any annexes to the international preliminary examination report. It is the applicant's responsibility to prepare and furnish such translation directly to each elected Office concerned.

For further details on the applicable time limits and requirements of the elected Offices, see Volume II of the PCT Applicant's Guide.

Name and mailing address of the IPEA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer
GEORGE DAVIS

Telephone No. (703) 305-3891

Peggy Hance

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

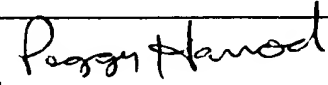
(PCT Article 36 and Rule 70)

Applicant's or agent's file reference 101/1	FOR FURTHER ACTION See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)	
International application No. PCT/IL00/00178	International filing date (<i>day/month/year</i>) 20 MARCH 2000	Priority date (<i>day/month/year</i>) 22 MARCH 1999
International Patent Classification (IPC) or national classification and IPC IPC(7): G06F 17/30 and US Cl.: 707/2		
Applicant INTERLINK COMPUTER COMMUNICATIONS LTD.		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.
2. This REPORT consists of a total of 3 sheets.
☒ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority. (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).
 These annexes consist of a total of 6 sheets.

3. This report contains indications relating to the following items:

- I ☒ Basis of the report
- II ☐ Priority
- III ☐ Non-establishment of report with regard to novelty, inventive step or industrial applicability
- IV ☐ Lack of unity of invention
- V ☒ Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- VI ☐ Certain documents cited
- VII ☐ Certain defects in the international application
- VIII ☐ Certain observations on the international application

Date of submission of the demand 20 OCTOBER 2000	Date of completion of this report 12 JUNE 2001
Name and mailing address of the IPEA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer  GEORGE DAVIS
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9703

I. Basis of the report**1. With regard to the elements of the international application:***☐ the international application as originally filed☒ the description:

pages (See Attached) _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____

☒ the claims:

pages (See Attached) _____, as originally filed
pages _____, as amended (together with any statement) under Article 19
pages _____, filed with the demand
pages _____, filed with the letter of _____

☒ the drawings:

pages (See Attached) _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____

☒ the sequence listing part of the description:

pages (See Attached) _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____

2. With regard to the language, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.

These elements were available or furnished to this Authority in the following language _____ which is:

☐ the language of a translation furnished for the purposes of international search (under Rule 23.1(b)).☐ the language of publication of the international application (under Rule 48.3(b)).☐ the language of the translation furnished for the purposes of international preliminary examination (under Rules 55.2 and/or 55.3).**3. With regard to any nucleotide and/or amino acid sequence disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:**☐ contained in the international application in printed form.☐ filed together with the international application in computer readable form.☐ furnished subsequently to this Authority in written form.☐ furnished subsequently to this Authority in computer readable form.☐ The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.☐ The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.**4. ☒ The amendments have resulted in the cancellation of:**☒ the description, pages NONE☒ the claims, Nos. NONE☒ the drawings, sheets/fig NONE**5. ☐ This report has been drawn as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).****

* Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17).

**Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**1. statement**

Novelty (N)	Claims	<u>NONE</u>	YES
	Claims	<u>1-30</u>	NO
Inventive Step (IS)	Claims	<u>NONE</u>	YES
	Claims	<u>1-30</u>	NO
Industrial Applicability (IA)	Claims	<u>NONE</u>	YES
	Claims	<u>1-30</u>	NO

2. citations and explanations (Rule 70.7)

Claims 1-30 lack novelty under PCT Article 33(2) as being anticipated by Coyler's "Bridge for a Client-Server Environment."

The claims are directed to a standard tool-of-the-trade known generally as a "wrapper." The specific implementation claimed is known more specifically as an "Object Request Broker" (ORB). Coyler's Figure 1 shows the use of ORBs in a client-server environment, as claimed by Applicant.

Claims 1-30 lack an inventive step under PCT Article 33(3) as being obvious over an ORB, as taught by Coyler. Applicant's claims relate to a standard tool-of-the-trade in database application development, as taught by Coyler.

Claims 1-30 lack industrial applicability as defined by PCT Article 33(4). Claims 1-30 are recite abstract ideas of writing a computer program without any limitation to apractical application. As such, the claimed invention is clearly ineligible for patent protection.

Response to Applicant Remarks filed March 06 March 2001:

Applicant argues at page 8, last paragraph that "automatically constructing the software program is not known in the art". However, the phrase "automatically constructing the software program" is in the preamble of claims 1 and 19 not in the items of these claims.

Applicant also argues at page 3, last paragraph that "the software program itself has industrial applicability. Applicant is in error since the computer program by itself has no industrial applicability.

Applicant further argues at page 4, lines 1-3 that "automatic construction of such a software program" in claims 1-30 has industrial applicability. First, automatic construction of software program does not indicate that the claim invention has (Continued on Supplemental Sheet.)

Supplemental Box

(To be used when the space in any of the preceding boxes is not sufficient)

Continuation of: Boxes I - VIII

Sheet 10

I. BASIS OF REPORT:

This report has been drawn on the basis of the description,
page(s) 1-15, as originally filed.
page(s) NONE, filed with the demand.
and additional amendments:
NONE

This report has been drawn on the basis of the claims,
page(s) NONE, as originally filed.
page(s) NONE, as amended under Article 19.
page(s) NONE, filed with the demand.
and additional amendments:
16-21, filed with the letter of 06 MARCH 2001

This report has been drawn on the basis of the drawings,
page(s) NONE, as originally filed.
page(s) 1-2, filed with the demand.
and additional amendments:
NONE

This report has been drawn on the basis of the sequence listing part of the description:
page(s) NONE, as originally filed.
pages(s) NONE, filed with the demand.
and additional amendments:
NONE

V. 2. REASONED STATEMENTS - CITATIONS AND EXPLANATIONS (Continued):

industrial applicability. Second, the "automatic construction of such a software program" is in the preamble of the claims, it is not in the items of claims 1 and 19, such as items (a)-(d).

----- NEW CITATIONS -----

NONE

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

REC'D 20 JUL 2001

WIPO PCT

Applicant's or agent's file reference I01/1	FOR FURTHER ACTION See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)	
International application No. PCT/IL00/00178	International filing date (day/month/year) 20 MARCH 2000	Priority date (day/month/year) 22 MARCH 1999
International Patent Classification (IPC) or national classification and IPC IPC(7): G06F 17/30 and US Cl.: 707/2		
Applicant INTERLINK COMPUTER COMMUNICATIONS LTD.		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.
2. This REPORT consists of a total of 3 sheets.

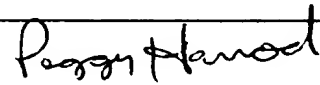
☒ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority. (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

These annexes consist of a total of 6 sheets.

3. This report contains indications relating to the following items:

- I ☒ Basis of the report
- II ☐ Priority
- III ☐ Non-establishment of report with regard to novelty, inventive step or industrial applicability
- IV ☐ Lack of unity of invention
- V ☒ Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- VI ☐ Certain documents cited
- VII ☐ Certain defects in the international application
- VIII ☐ Certain observations on the international application

**CORRECTED
VERSION**

Date of submission of the demand 20 OCTOBER 2000	Date of completion of this report 12 JUNE 2001
Name and mailing address of the IPEA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer  GEORGE DAVIS
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9703

I. Basis of the report**1. With regard to the elements of the international application:***

- ☐ the international application as originally filed
- ☒ the description:
pages (See Attached) _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____
- ☒ the claims:
pages (See Attached) _____, as originally filed
pages _____, as amended (together with any statement) under Article 19
pages _____, filed with the demand
pages _____, filed with the letter of _____
- ☒ the drawings:
pages (See Attached) _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____
- ☒ the sequence listing part of the description:
pages (See Attached) _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____

2. With regard to the language, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.

These elements were available or furnished to this Authority in the following language _____ which is:

- ☐ the language of a translation furnished for the purposes of international search (under Rule 23.1(b)).
- ☐ the language of publication of the international application (under Rule 48.3(b)).
- ☐ the language of the translation furnished for the purposes of international preliminary examination (under Rules 55.2 and/or 55.3).

3. With regard to any nucleotide and/or amino acid sequence disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:

- ☐ contained in the international application in printed form.
- ☐ filed together with the international application in computer readable form.
- ☐ furnished subsequently to this Authority in written form.
- ☐ furnished subsequently to this Authority in computer readable form.
- ☐ The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
- ☐ The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.

4. ☒ The amendments have resulted in the cancellation of:

- ☒ the description, pages NONE
- ☒ the claims, Nos. NONE
- ☒ the drawings, sheets/fig NONE

5. ☐ This report has been drawn as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).**

** Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17).*

***Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.*

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. statement

Novelty (N)	Claims	<u>NONE</u>	YES
	Claims	<u>1-30</u>	NO
Inventive Step (IS)	Claims	<u>NONE</u>	YES
	Claims	<u>1-30</u>	NO
Industrial Applicability (IA)	Claims	<u>NONE</u>	YES
	Claims	<u>1-30</u>	NO

2. citations and explanations (Rule 70.7)

Claims 1-30 lack novelty under PCT Article 33(2) as being anticipated by Coyler's "Bridge for a Client-Server Environment."

The claims are directed to a standard tool-of-the-trade known generally as a "wrapper." The specific implementation claimed is known more specifically as an "Object Request Broker" (ORB). Coyler's Figure 1 shows the use of ORBs in a client-server environment, as claimed by Applicant.

Claims 1-30 lack an inventive step under PCT Article 33(3) as being obvious over an ORB, as taught by Coyler. Applicant's claims relate to a standard tool-of-the-trade in database application development, as taught by Coyler.

Claims 1-30 lack industrial applicability as defined by PCT Article 33(4). Claims 1-30 are recite abstract ideas of writing a computer program without any limitation to a practical application. As such, the claimed invention is clearly ineligible for patent protection.

Response to Applicant Remarks filed March 06 March 2001:

Applicant argues at page 8, last paragraph that "automatically constructing the software program is not known in the art". However, the phrase "automatically constructing the software program" is in the preamble of claims 1 and 19 not in the items of these claims.

Applicant also argues at page 3, last paragraph that "the software program itself has industrial applicability. Applicant is in error since the computer program by itself has no industrial applicability.

Applicant further argues at page 4, lines 1-3 that "automatic construction of such a software program" in claims 1-30 has industrial applicability. First, automatic construction of software program does not indicate that the claim invention has
(Continued on Supplemental Sheet.)

Supplemental Box

(To be used when the space in any of the preceding boxes is not sufficient)

Continuation of: Boxes I - VIII

Sheet 10

I. BASIS OF REPORT:

This report has been drawn on the basis of the description,
page(s) 1-15, as originally filed.
page(s) NONE, filed with the demand.
and additional amendments:
NONE

This report has been drawn on the basis of the claims,
page(s) NONE, as originally filed.
page(s) NONE, as amended under Article 19.
page(s) NONE, filed with the demand.
and additional amendments:
16-21, filed with the letter of 06 MARCH 2001

This report has been drawn on the basis of the drawings,
page(s) NONE, as originally filed.
page(s) 1-2, filed with the demand.
and additional amendments:
NONE

This report has been drawn on the basis of the sequence listing part of the description:
page(s) NONE, as originally filed.
pages(s) NONE, filed with the demand.
and additional amendments:
NONE

V. 2. REASONED STATEMENTS - CITATIONS AND EXPLANATIONS (Continued):

industrial applicability. Second, the "automatic construction of such a software program" is in the preamble of the claims, it is not in the items of claims 1 and 19, such as items (a)-(d).

----- NEW CITATIONS -----
NONE

WHAT IS CLAIMED IS:

1. A system for automatic construction of a software program to write data to, and to retrieve data from, an enterprise resource for interaction with a user, the system comprising:
 - (a) a meta-data description of the enterprise resource, said meta-data description featuring at least one attribute of the data of the enterprise resource, said at least one attribute including a structure of the data;
 - (b) a meta-data parser for parsing the enterprise resource into said plurality of logical units of data according to said at least one attribute of said meta-data description, and for determining a hierarchical structure for said logical units of data according to said structure of the data;
 - (c) a meta-data translator for translating each logical unit of data into an application component, such that the data of said logical unit of data is associated with said application component, and for associating said application component with at least one interface component for interfacing with the enterprise resource; and
 - (d) a definition extraction factory for creating the software program from a plurality of said application components with said at least one interface component, at least according to said hierarchical structure for said plurality of logical units of data.
2. The system of claim 1, wherein the software program features a plurality of interface components, including at least one interface component for data flow to and from the enterprise resource and at least one interface component for providing a user interface to the user.
3. The system of claim 2, wherein the software program further comprises:
 - (i) a first layer of interface components for providing control of the software program, including at least one interface component for controlling said user interface and at least one interface component for controlling said data flow;

- (ii) a second layer of interface components for providing said data input and said data output, including said at least one interface component for providing said user interface; and
 - (iii) a third layer of interface components for specifically interacting with the enterprise resource.
4. The system of claim 3, wherein said application component is contained in said third layer and said third layer further comprises:
- (1) an export interface for retrieving data from the enterprise resource; and
 - (2) an import interface for storing data into the enterprise resource.
5. The system of claim 4, wherein each of said first layer and said second layer includes an extension of said application component for communicating with said at least one interface component related to said user interface and said at least one interface component related to said data flow.
6. The system of claim 5, wherein said at least one interface component is defined according to a standard interface and according to at least one user-defined function.
7. The system of claim 1, wherein said meta-data description, said meta-data parser and said meta-data translator are specific for a particular type of enterprise resource.
8. The system of claim 7, wherein said meta-data translator translates each logical unit of data according to at least one internal translation rule.
9. The system of claim 8, wherein said at least one internal translation rule is defined according to an attribute of said particular type of enterprise resource.
10. The system of claim 9, wherein said at least one internal translation rule further includes at least one default value for a parameter not described in said meta-data description.

IPEA/US 26 MAR 2001

11. The system of claim 10, wherein said meta-data translator also translates each logical unit of data according to at least one external translation rule defined by the user.

12. The system of claim 1, further comprising:

- (e) an application component factory for transforming each application component into an object having said at least one method and being associated with the data of said associated logical data unit; and
- (f) a hierarchical structure constructor for arranging said plurality of application components into a hierarchical object-oriented structure according to said hierarchical structure for said logical units of data, such that said definition extraction factory also creates the software program according to said hierarchical object-oriented structure.

13. The system of claim 12, further comprising:

- (g) a definitions file for defining each of said plurality of application components, such that the software program is stored as said definitions file, said definitions file being constructed by said definition extraction factory.

14. The system of claim 13, further comprising:

- (h) an editing environment for editing said hierarchical object-oriented structure by the user, such that the user manually alters at least a portion of said hierarchical object oriented structure.

15. The system of claim 14, further comprising:

- (i) a parsing engine for reading said definitions file and for operating the software program according to said definitions file.

16. The system of claim 15, further comprising:

- (j) a user interface of the software program for interacting with the user;

and

AMENDED SHEET

IPEA/US 26 MAR 2001

- (k) a client computer for operating said user interface, such that said parsing engine constructs said user interface automatically according to at least one property of said client computer.

17. The system of claim 16, wherein said user interface is a GUI (graphical user interface), said GUI being constructed by said meta-data translator with at least one abstract GUI component, the system further comprising:

- (l) at least one GUI library for containing a conversion of said at least one abstract GUI component to a concrete GUI component, such that said parsing engine also constructs said user interface automatically according to said GUI library.

18. The system of claim 17, wherein said at least one GUI library contains a plurality of concrete GUI components and a layout manager for determining a spatial relationship between said plurality of concrete GUI components.

19. A method for automatically constructing a software program to write data to, and retrieve data from, an enterprise data resource for a user, the steps of the method being performed by a data processor, the method comprising the steps of:

- (a) providing a meta-data description of the enterprise resource, said meta-data description including at least one attribute of the data of the enterprise resource;
- (b) dividing the enterprise resource into a plurality of logical data units according to said meta-data description;
- (c) translating each of said plurality of logical data units into an application component according to at least one attribute of said meta-data description to form a plurality of application components;
- (d) determining a hierarchical structure for said plurality of logical data units; and
- (e) constructing the software program from said plurality of application components according to said hierarchical structure for said plurality of logical data units.

20. The method of claim 19, wherein said at least one attribute of said

AMENDED SHEET

IPEA/US 26 MAR 2001

meta-data description includes a structure of the data in the enterprise resource, such that step (d) is performed according to said structure of the data in the enterprise resource.

21. The method of claim 20, wherein step (c) further comprises the step of:
 - (i) defining at least one method for each application component, for operating on the data of each logical data unit associated with each application component, such that each application component is an object having said at least one method and being associated with the data of said associated logical data unit.
22. The method of claim 21, wherein step (d) further comprises the step of:
 - (i) constructing a hierarchical object-oriented structure for said plurality of application components according to said hierarchical structure for said plurality of logical data units.
23. The method of claim 22, wherein step (d) further comprises the steps of:
 - (ii) providing an editing environment for displaying said hierarchical object-oriented structure to the user; and
 - (iii) altering at least a portion of said hierarchical object-oriented structure by the user.
24. The method of claim 23, wherein step (c) is performed according to at least one internal translation rule, said at least one internal translation rule being defined according to a type of the enterprise resource.
25. The method of claim 24, wherein step (c) is additionally performed according to at least one external translation rule defined by the user and wherein step (i) of step (c) is performed by translating each of said plurality of logical data units into an application component according to said at least one external translation rule.
26. The method of claim 25, further comprising the steps of:

AMENDED SHEET

- (f) extracting a plurality of definitions for describing each of said plurality of application components and said hierarchical structure; and
 - (g) storing the software program by storing said plurality of definitions.
27. The method of claim 26, further comprising the step of:
- (h) operating the software program according to said plurality of definitions.
28. The method of claim 27, further comprising the step of:
- (i) altering the software program by changing at least one of said plurality of definitions.
29. The method of claim 24, wherein the software program is operated by a client computer, and step (e) includes the steps of:
- (1) automatically constructing a user interface according to at least one property of said client computer; and
 - (2) displaying said user interface to the user for interacting with the user.
30. The method of claim 29, wherein step (1) further includes the steps of:
- (A) constructing an abstract user interface for displaying data from the enterprise resource and for interacting with the user; and
 - (B) constructing a concrete user interface according to said abstract user interface and according to said at least one property of said client computer.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL00/00178

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/30

US CL : 707/2

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/2

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,862,328 A (COLYER) 19 January 1999, ALL	1-28

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
A document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance, the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
E earlier document published on or after the international filing date	*Y* document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	*A* document member of the same patent family
O document referring to an oral disclosure, use, exhibition or other means	
P document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

07 JUNE 2000

Date of mailing of the international search report

06 JUL 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

TARIQ HAFIZ

Telephone No. (703) 305-9643

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL00/00178

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☒ Claims Nos.: 1-28
because they relate to subject matter not required to be searched by this Authority, namely:
Please See Extra Sheet.
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest.
☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL00/00178

BOX I. OBSERVATIONS WHERE CLAIMS WERE FOUND UNSEARCHABLE

1. Subject matter not required to be searched by this ISA, namely:

The subject matter of the claimed system (claims 1-18) consists of a data structure, and a method for manipulating the data structure without limitation to a practical application. As such, the claimed invention merely cites functional descriptive material, which is ineligible for patent protection under 35 U.S.C. 101.

The subject matter of the claimed method (claims 19-28) similarly consists of manipulating a data structure without limitation to a practical application. Thus, claims 19-28 are also ineligible for patent protection under 35 U.S.C. 101.

September 20, 2001

Carol Platt
Senior Legal Assistant, Intellectual Property
Marvel Entertainment Group, Inc.
387 Park Avenue South
New York, NY 10016

Re: Israel Trademark Registration No. 89763 for **MARVEL COMICS LOGO** in
class 25 on behalf of Marvel Characters, Inc.
Our ref.: 003-004L-240

Dear Ms. Platt,

Further to our fax of September 3, 2001, I am pleased to advise you that the Israel Trademark Office has amended the Hebrew specification of goods. The Certificate of Registration is enclosed. Please sign and return (by mail or facsimile) the enclosed copy of this letter to acknowledge that you have safely received the enclosed Certificate.

Because the mark was filed over seven (7) years ago, November 9, 1993, it is due for renewal for a period of fourteen (14) years. In order to avoid late renewal fees, I look forward to your earliest authorization to pay the renewal fee for this registration.

If at any time this mark is altered or the range of goods extended, I recommend that you consult with me to ensure that the registration provides adequate protection.

Please also contact me if there is a change in the Registrant's name or address or if the trademark is licensed or assigned, as these may need to be recorded on the Register.

I remind you that this registration will become vulnerable to cancellation for non-use if the mark is not used in relation to goods covered by the registration for three (3) consecutive years preceding the date of a third party's claim for cancellation on the ground of non-use.

Israel currently operates under the registered user system to prove use of a trademark by someone other than the registrant. Under Israel trademark law, if a registrant licenses a third party to use a trademark or service mark in Israel (other than as a mere distributor), that third party must be recorded as a registered user to assure that the trademark's use will inure to the benefit of the registrant.

Ms. Carol Platt
September 20, 2001
Page Two

Under strict Israeli trademark practice, failure to record a registered user may result in the registration becoming vulnerable to cancellation for non-use by third parties.

If you wish further information on recording a registered user, please contact me.

It is not mandatory in Israel to indicate that the mark is registered. However, it is advisable to do so.

I take this opportunity to enclose our final debit note for services rendered in this matter.

Thank you for your instructions.

Sincerely,

David Wolberg
Plinner, Bodner & Co.

encls
edsw5319

PCT

REQUEST

The undersigned requests that the present international application be processed according to the Patent Cooperation Treaty.

For receiving Office use only

International Application No.

International Filing Date

Name of receiving Office and "PCT International Application"

Applicant's or agent's file reference
(if desired) (12 characters maximum) I01/1

Box No. I TITLE OF INVENTION

AUTOMATIC INTERFACE GENERATION FOR AN ENTERPRISE RESOURCE

Box No. II APPLICANT

Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)

INTERLINK COMPUTER COMMUNICATIONS LTD.
5 JABOTINSKY STREET
RAMAT-GAN
ISRAEL

☐ This person is also inventor.

Telephone No.
972-3-576-6973

Facsimile No.
972-3-751-3626

Teleprinter No.

State (that is, country) of nationality:
IL

State (that is, country) of residence:
IL

This person is applicant for the purposes of: ☐ all designated States ☒ all designated States except the United States of America ☐ the United States of America only ☐ the States indicated in the Supplemental Box

Box No. III FURTHER APPLICANT(S) AND/OR (FURTHER) INVENTOR(S)

Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)

HADAS ILAN
HAHADARIM 127
TZORAN
ISRAEL

This person is:

☐ applicant only

☒ applicant and inventor

☐ inventor only (If this check-box is marked, do not fill in below.)

State (that is, country) of nationality:
IL

State (that is, country) of residence:
IL

This person is applicant for the purposes of: ☐ all designated States ☐ all designated States except the United States of America ☒ the United States of America only ☐ the States indicated in the Supplemental Box

☐ Further applicants and/or (further) inventors are indicated on a continuation sheet.

Box No. IV AGENT OR COMMON REPRESENTATIVE; OR ADDRESS FOR CORRESPONDENCE

The person identified below is hereby/has been appointed to act on behalf of the applicant(s) before the competent International Authorities as:

☒ agent

☐ common representative

Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country.)

PLINNER SA'AR, ADV.
10 DIZENGOFF STREET
TEL AVIV
ISRAEL

Telephone No.
972-3-620-1866

Facsimile No.
972-3-620-1469

Teleprinter No.

☐ Address for correspondence: Mark this check-box where no agent or common representative is/has been appointed and the space above is used instead to indicate a special address to which correspondence should be sent.

Box No.V DESIGNATION OF STATES

The following designations are hereby made under Rule 4.9(a) (mark the applicable check-boxes; at least one must be marked):

Regional Patent

- ☒ **AP ARIPO Patent:** GH Ghana, GM Gambia, KE Kenya, LS Lesotho, MW Malawi, SD Sudan, SL Sierra Leone, SZ Swaziland, UG Uganda, ZW Zimbabwe, and any other State which is a Contracting State of the Harare Protocol and of the PCT
- ☒ **EA Eurasian Patent:** AM Armenia, AZ Azerbaijan, BY Belarus, KG Kyrgyzstan, KZ Kazakhstan, MD Republic of Moldova, RU Russian Federation, TJ Tajikistan, TM Turkmenistan, and any other State which is a Contracting State of the Eurasian Patent Convention and of the PCT
- ☒ **EP European Patent:** AT Austria, BE Belgium, CH and LI Switzerland and Liechtenstein, CY Cyprus, DE Germany, DK Denmark, ES Spain, FI Finland, FR France, GB United Kingdom, GR Greece, IE Ireland, IT Italy, LU Luxembourg, MC Monaco, NL Netherlands, PT Portugal, SE Sweden, and any other State which is a Contracting State of the European Patent Convention and of the PCT
- ☒ **OA OAPI Patent:** BF Burkina Faso, BJ Benin, CF Central African Republic, CG Congo, CI Côte d'Ivoire, CM Cameroon, GA Gabon, GN Guinea, GW Guinea-Bissau, ML Mali, MR Mauritania, NE Niger, SN Senegal, TD Chad, TG Togo, and any other State which is a member State of OAPI and a Contracting State of the PCT (if other kind of protection or treatment desired, specify on dotted line)

National Patent (if other kind of protection or treatment desired, specify on dotted line):

- | | |
|--|--|
| <input checked="" type="checkbox"/> AE United Arab Emirates | <input checked="" type="checkbox"/> LR Liberia |
| <input checked="" type="checkbox"/> AL Albania | <input checked="" type="checkbox"/> LS Lesotho |
| <input checked="" type="checkbox"/> AM Armenia | <input checked="" type="checkbox"/> LT Lithuania |
| <input checked="" type="checkbox"/> AT Austria | <input checked="" type="checkbox"/> LU Luxembourg |
| <input checked="" type="checkbox"/> AU Australia | <input checked="" type="checkbox"/> LV Latvia |
| <input checked="" type="checkbox"/> AZ Azerbaijan | <input checked="" type="checkbox"/> MD Republic of Moldova |
| <input checked="" type="checkbox"/> BA Bosnia and Herzegovina | <input checked="" type="checkbox"/> MG Madagascar |
| <input checked="" type="checkbox"/> BB Barbados | <input checked="" type="checkbox"/> MK The former Yugoslav Republic of Macedonia |
| <input checked="" type="checkbox"/> BG Bulgaria | |
| <input checked="" type="checkbox"/> BR Brazil | <input checked="" type="checkbox"/> MN Mongolia |
| <input checked="" type="checkbox"/> BY Belarus | <input checked="" type="checkbox"/> MW Malawi |
| <input checked="" type="checkbox"/> CA Canada | <input checked="" type="checkbox"/> MX Mexico |
| <input checked="" type="checkbox"/> CH and LI Switzerland and Liechtenstein | <input checked="" type="checkbox"/> NO Norway |
| <input checked="" type="checkbox"/> CN China | <input checked="" type="checkbox"/> NZ New Zealand |
| <input checked="" type="checkbox"/> CU Cuba | <input checked="" type="checkbox"/> PL Poland |
| <input checked="" type="checkbox"/> CZ Czech Republic | <input checked="" type="checkbox"/> PT Portugal |
| <input checked="" type="checkbox"/> DE Germany | <input checked="" type="checkbox"/> RO Romania |
| <input checked="" type="checkbox"/> DK Denmark | <input checked="" type="checkbox"/> RU Russian Federation |
| <input checked="" type="checkbox"/> EE Estonia | <input checked="" type="checkbox"/> SD Sudan |
| <input checked="" type="checkbox"/> ES Spain | <input checked="" type="checkbox"/> SE Sweden |
| <input checked="" type="checkbox"/> FI Finland | <input checked="" type="checkbox"/> SG Singapore |
| <input checked="" type="checkbox"/> GB United Kingdom | <input checked="" type="checkbox"/> SI Slovenia |
| <input checked="" type="checkbox"/> GD Grenada | <input checked="" type="checkbox"/> SK Slovakia |
| <input checked="" type="checkbox"/> GE Georgia | <input checked="" type="checkbox"/> SL Sierra Leone |
| <input checked="" type="checkbox"/> GH Ghana | <input checked="" type="checkbox"/> TJ Tajikistan |
| <input checked="" type="checkbox"/> GM Gambia | <input checked="" type="checkbox"/> TM Turkmenistan |
| <input checked="" type="checkbox"/> HR Croatia | <input checked="" type="checkbox"/> TR Turkey |
| <input checked="" type="checkbox"/> HU Hungary | <input checked="" type="checkbox"/> TT Trinidad and Tobago |
| <input checked="" type="checkbox"/> ID Indonesia | <input checked="" type="checkbox"/> UA Ukraine |
| <input checked="" type="checkbox"/> IL Israel | <input checked="" type="checkbox"/> UG Uganda |
| <input checked="" type="checkbox"/> IN India | <input checked="" type="checkbox"/> US United States of America |
| <input checked="" type="checkbox"/> IS Iceland | |
| <input checked="" type="checkbox"/> JP Japan | <input checked="" type="checkbox"/> UZ Uzbekistan |
| <input checked="" type="checkbox"/> KE Kenya | <input checked="" type="checkbox"/> VN Viet Nam |
| <input checked="" type="checkbox"/> KG Kyrgyzstan | <input checked="" type="checkbox"/> YU Yugoslavia |
| <input checked="" type="checkbox"/> KP Democratic People's Republic of Korea | <input checked="" type="checkbox"/> ZA South Africa |
| | <input checked="" type="checkbox"/> ZW Zimbabwe |
| <input checked="" type="checkbox"/> KR Republic of Korea | Check-boxes reserved for designating States which have become party to the PCT after issuance of this sheet: |
| <input checked="" type="checkbox"/> KZ Kazakhstan | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> LC Saint Lucia | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> LK Sri Lanka | <input checked="" type="checkbox"/> |

Precautionary Designation Statement: In addition to the designations made above, the applicant also makes under Rule 4.9(b) all other designations which would be permitted under the PCT except any designation(s) indicated in the Supplemental Box as being excluded from the scope of this statement. The applicant declares that those additional designations are subject to confirmation and that any designation which is not confirmed before the expiration of 15 months from the priority date is to be regarded as withdrawn by the applicant at the expiration of that time limit. (Confirmation of a designation consists of the filing of a notice specifying that designation and the payment of the designation and confirmation fees. Confirmation must reach the receiving Office within the 15-month time limit.)

Box No. VI PRIORITY CLAIM		<input type="checkbox"/> Further priority claims are indicated in the Supplemental Box.		
Filing date of earlier application (day/month/year)	Number of earlier application	Where earlier application is:		
		national application: country	regional application: regional Office	international application: receiving Office
item (1) 22/03/99	09/273,464	USA		
item (2)				
item (3)				
<input type="checkbox"/> The receiving Office is requested to prepare and transmit to the International Bureau a certified copy of the earlier application(s) (only if the earlier application was filed with the Office which for the purposes of the present international application is the receiving Office) identified above as item(s):				
<small>* Where the earlier application is an ARIPO application, it is mandatory to indicate in the Supplemental Box at least one country party to the Paris Convention for the Protection of Industrial Property for which that earlier application was filed (Rule 4.10(b)(ii)). See Supplemental Box.</small>				
Box No. VII INTERNATIONAL SEARCHING AUTHORITY				
Choice of International Searching Authority (ISA) <small>(if two or more International Searching Authorities are competent to carry out the international search, indicate the Authority chosen; the two-letter code may be used):</small>		Request to use results of earlier search; reference to that search (if an earlier search has been carried out by or requested from the International Searching Authority):		
ISA / US		Date (day/month/year) Number Country (or regional Office)		
Box No. VIII CHECK LIST; LANGUAGE OF FILING				
This international application contains the following number of sheets: request : 3 description (excluding sequence listing part) : 15 claims : 5 abstract : 1 drawings : 2 sequence listing part of description : 0 Total number of sheets : 26		This international application is accompanied by the item(s) marked below: 1. <input checked="" type="checkbox"/> fee calculation sheet 2. <input type="checkbox"/> separate signed power of attorney 3. <input type="checkbox"/> copy of general power of attorney; reference number, if any: 4. <input type="checkbox"/> statement explaining lack of signature 5. <input type="checkbox"/> priority document(s) identified in Box No. VI as item(s): 6. <input type="checkbox"/> translation of international application into (language): 7. <input type="checkbox"/> separate indications concerning deposited microorganism or other biological material 8. <input type="checkbox"/> nucleotide and/or amino acid sequence listing in computer readable form 9. <input type="checkbox"/> other (specify):		
Figure of the drawings which should accompany the abstract: 1		Language of filing of the international application: ENGLISH		
Box No. IX SIGNATURE OF APPLICANT OR AGENT				
<small>Next to each signature, indicate the name of the person signing and the capacity in which the person signs (if such capacity is not obvious from reading the request).</small>				
 PLINNER SA'AR, AGENT סער פלינר, עו"ד 10 DIZENGOFF STREET SA'AR PLINNER, Adv. TEL AVIV ISRAEL				

For receiving Office use only		2. Drawings: <input type="checkbox"/> received: <input type="checkbox"/> not received:
1. Date of actual receipt of the purported international application:		
3. Corrected date of actual receipt due to later but timely received papers or drawings completing the purported international application:		
4. Date of timely receipt of the required corrections under PCT Article 11(2):		
5. International Searching Authority (if two or more are competent): ISA /		6. <input type="checkbox"/> Transmittal of search copy delayed until search fee is paid.

For International Bureau use only
Date of receipt of the record copy by the International Bureau:

This sheet is not part of and does not count as a sheet of the international application.

PCT

FEE CALCULATION SHEET

Annex to the Request

For receiving Office use only

International application No.

Applicant's or agent's
file reference 101/1

Date stamp of the receiving Office

Applicant

INTERLINK COMPUTER COMMUNICATIONS LTD.

CALCULATION OF PRESCRIBED FEES

1. TRANSMITTAL FEE

137 sheets T

2. SEARCH FEE

\$700 S

International search to be carried out by

US

(If two or more International Searching Authorities are competent in relation to the international application, indicate the name of the Authority which is chosen to carry out the international search.)

3. INTERNATIONAL FEE

Basic Fee

The international application contains 26 sheets.

first 30 sheets

0

x

10

\$427 b1

remaining sheets

additional amount

0 b2

Add amounts entered at b1 and b2 and enter total at B

\$427 B

Designation Fees

The international application contains 11 designations.

10

x

number of designation fees
payable (maximum 10)

amount of designation fee

\$886 D

Add amounts entered at B and D and enter total at I

(Applicants from certain States are entitled to a reduction of 75% of the international fee. Where the applicant is (or all applicants are) so entitled, the total to be entered at I is 25% of the sum of the amounts entered at B and D.)

\$1263 I

4. FEE FOR PRIORITY DOCUMENT (if applicable)

0 P

5. TOTAL FEES PAYABLE

Add amounts entered at T, S, I and P, and enter total in the TOTAL box

137 sheets / \$1968

TOTAL

☐ The designation fees are not paid at this time.

MODE OF PAYMENT

☐ authorization to charge
deposit account (see below)

☐ bank draft

☐ coupons

☐ cheque

☐ cash

☐ other (specify):

☒ postal money order

☐ revenue stamps

DEPOSIT ACCOUNT AUTHORIZATION (this mode of payment may not be available at all receiving Offices)

The RO/ ☐ is hereby authorized to charge the total fees indicated above to my deposit account.

☐ (this check-box may be marked only if the conditions for deposit accounts of the receiving Office so permit) is hereby authorized to charge any deficiency or credit any overpayment in the total fees indicated above to my deposit account.

☐ is hereby authorized to charge the fee for preparation and transmittal of the priority document to the International Bureau of WIPO to my deposit account.

Deposit Account No.

Date (day/month/year)

Signature

2/p.17

AUTOMATIC INTERFACE GENERATION FOR AN
ENTERPRISE RESOURCE**FIELD AND BACKGROUND OF THE INVENTION**

5 The present invention relates to a system and method for automatically generating an application for interacting with an enterprise resource, and in particular, for such a system and method which constructs such an application without extensive manual intervention from a human programmer for an enterprise resource, such that the data associated with the enterprise resource is available through the application, for manipulation, export and import of data.

10 An enterprise resource is a structured set of data which is accessible through an enterprise application, for example by manipulating, exporting and importing data. The enterprise application is a software program which enables such access and manipulation of the data of the resource. One example of an enterprise resource is a database which is accessible through an interface written in the COBOL programming language. The term "enterprise"
15 refers to an existing resource and application within an organization, for example. Such enterprise resources may be maintained simply because constructing a completely new data structure and a new application interface is expensive and time-consuming, and may also result in significant "down time" until the new application is running smoothly and the enterprise application can be removed. Thus, organizations are hesitant to completely replace enterprise
20 applications and resources.

A more useful system and method would enable the organization to integrate new or additional functionality to the enterprise application and resource, while still maintaining the enterprise application and resource, without requiring extensive programming by a human programmer and without requiring extensive testing before the new application and data
25 structure is ready for use. Such a system and method would be either semi-automatic, or even completely automatic, thereby reducing the possibility of human error as well as reducing the amount of human intervention required to prepare the new application and resource. Finally, such a system and method would produce an application which is widely usable across many different computer platforms, yet which is flexible and robust. Unfortunately, such a system
30 and method are not currently available.

There is thus a need for, and it would be useful to have, a system and a method for automatic generation of a new resource data structure and of a software interface for that data structure, which would interface with an enterprise resource and an enterprise application, and

which would be flexible and robust, yet would not require substantial manual intervention by a human programmer for the construction of the new data structure and software interface, thereby integrating the enterprise application or applications into a single client interface.

5 **BRIEF DESCRIPTION OF THE DRAWINGS**

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, wherein:

FIG. 1 is a schematic block diagram of an illustrative system and work flow according
10 to the present invention; and

FIG. 2 is a schematic block diagram of an application created according to the present invention.

SUMMARY OF THE INVENTION

15 The present invention is of a system and a method for automatic creation of a software application for accessing an enterprise data resource, preferably without actually altering the stored data. Rather, the system and method of the present invention create an interface which is preferably operable across computer platforms, and which enables the user to both write data to, and retrieve data from the enterprise resource. This is accomplished by first analyzing the
20 enterprise resource to decompose it into a plurality of logical data units. Each data unit is then translated into an application component, which is preferably an object with associated methods and data. Each application component is then added to a hierarchical object-oriented structure, which is constructed according to the logical relationships between the units of data. The hierarchical object-oriented structure is optionally edited manually by the user. Finally, the
25 hierarchical object-oriented structure is transformed into the application by an engine which uses a "factory" model, thereby efficiently coding those portions of the interface which are similar or even identical for all enterprise resources. The final application is preferably a group of objects which provide such functions as a user interface and data input and output.

According to the present invention, there is provided a system for automatic
30 construction of an application for an enterprise resource for operation by a user, the system comprising: (a) a meta-data description of the enterprise resource, the meta-data description featuring a plurality of logical units of data; (b) a meta-data parser for parsing the enterprise resource into the plurality of logical units of data according to the meta-data description and for

determining a relationship between the logical units of data; (c) a meta-data translator for translating each logical unit of data into an application component, for associating the application component with at least one interface component for interfacing with the enterprise resource; and (d) a definition extraction factory for creating the application from a plurality of the application components with the at least one interface component, at least according to the relationship between the plurality of logical units of data.

According to another embodiment of the present invention, there is provided a method for automatically constructing an application for an enterprise data resource for a user, the steps of the method being performed by a data processor, the method comprising the steps of:

(a) providing a meta-data description of the enterprise resource, the meta-data description including at least one attribute of the enterprise resource; (b) dividing the enterprise resource into a plurality of logical data units according to the meta-data description; (c) translating each of the plurality of logical data units into an application component according to at least one attribute of the meta-data description to form a plurality of application components; (d) determining a relationship between the plurality of logical data units; and (e) constructing the application according to the plurality of application components and the relationship between the plurality of logical data units.

Hereinafter, the term "network" refers to a connection between any two computers which permits the transmission of data. Hereinafter, the term "computer" includes, but is not limited to, personal computers (PC) having an operating system such as DOS, Windows™, OS/2™ or Linux; Macintosh™ computers; computers having any type of Java virtual machine as the operating system; and graphical workstations such as the computers of Sun Microsystems™ and Silicon Graphics™, and other computers having some version of the UNIX operating system such as AIX™ or SOLARIS™ of Sun Microsystems™; or any other known and available operating system, including operating systems such as Windows CE™ for embedded systems, including cellular telephones, handheld computational devices and palmtop computational devices, and any other computational device which can be connected to a network. Hereinafter, the term "Windows™" includes but is not limited to Windows95™, Windows 3.x™ in which "x" is an integer such as "1", Windows NT™, Windows98™, Windows CE™ and any upgraded versions of these operating systems by Microsoft Corp. (USA).

Hereinafter, the term "Web browser" refers to any software program, which can display multimedia data such as text, graphics, or both, from Web pages on World Wide Web sites.

Hereinafter, the term "Web page" refers to any document available for download and display, including, but not limited to, documents written in a mark-up language such as HTML (Hyper-text Mark-up Language), VRML (Virtual Reality Modeling Language), dynamic HTML, XML (Extended Mark-up Language) or related computer languages thereof.

5 Hereinafter, the term "application user" refers to the individual operating the application which is constructed according to the present invention, while the term "programming user" refers to the human programmer who is constructing the application by interacting with the system of the present invention.

10 The present invention could be described as a series of steps implemented by a data processor, such that the present invention could be implemented as hardware, software or firmware, or a combination thereof. For the present invention, a software application could be written in substantially suitable programming language, which could easily be selected by one of ordinary skill in the art. The programming language chosen should be compatible with the computer by which the software application is executed. Examples of suitable programming
15 languages include, but are not limited to, C, C++ and Java.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is of a system and a method for automatic creation of a software application for accessing an enterprise data resource, preferably without actually altering the
20 stored data. Rather, the system and method of the present invention create an application which is preferably operable across computer platforms, and which enables the application user to both write data to, and retrieve data from the enterprise resource. This is accomplished by first analyzing the enterprise resource to decompose it into a plurality of logical data units. Each data unit is then translated into an application component, which is preferably an object with
25 associated methods and data. Each application component is then added to a hierarchical structure which is preferably object-oriented. This hierarchical object-oriented structure is constructed according to the logical relationships between the units of data. By "object-oriented", it is meant that the hierarchical object-oriented structure features a plurality of objects, with data and methods, which are linked according to the relationships between
30 these objects. The hierarchical object-oriented structure is optionally edited manually by the programming user. Finally, the hierarchical object-oriented structure is transformed into the application by an engine which uses a "factory" model, thereby efficiently coding those portions of the interface which are similar or even identical for all enterprise resources. The

final application is preferably constructed as a group of component objects which provide such functions as a user interface and data input and output.

The final application is preferably adjusted to the requirements of a requesting client at run-time, particularly with regard to the GUI. For example, if the client which is requesting access to the enterprise resource data is a thin client operating a Web browser through HTML, then preferably the GUI is implemented as a Web page with HTML forms. More preferably, the GUI is first constructed as a GUI abstraction, thereby enabling different specific, concrete GUI implementations to be constructed "on the fly" as described in greater detail below.

The term "enterprise resource" includes such resources as a database or other data storage structure written in the programming language COBOL, for example, or a terminal emulator interface which enables the application user to access data through a particular type of terminal. Other examples of such enterprise resources are also possible within the scope of the present invention. However, for the purposes of discussion only, and without intending to be limiting in any way, the following description centers upon two types of enterprise resources: a data storage structure written in COBOL, and a terminal emulation software interface.

The principles and operation of a method and system according to the present invention may be better understood with reference to the drawings and the accompanying description, it being understood that these drawings are given for illustrative purposes only and are not meant to be limiting.

Referring now to the drawings, Figure 1 is a schematic block diagram of the workflow through the software modules in a system 10 for automatic generation of a new software application for an enterprise resource, preferably as a hierarchical object-oriented structure. A system 10 initially receives a meta-data description 12 of the data structure of the enterprise resource. This meta-data description divides the data of the enterprise resource into a plurality of logical data units 14. For example, for a data structure written in the COBOL programming language, each logical data unit 14 would be determined according to the syntax of the COBOL programming language. This syntax divides the data into different types, each of which is stored in a particular field. Thus, for a data structure written in COBOL, one example of a logical data unit 14 would be a particular type of data field.

As another example, for a terminal emulation software program such as a terminal emulation program for the 3270 terminal type, meta-data description 12 would include information obtained from trapping a plurality of screen displays for that terminal. Again, each screen display could be divided into data entry fields, for example, such that each data entry

field could be an example of logical data unit 14.

Meta-data description 12 for a particular enterprise resource is then received by a meta-data parser 16. Preferably, each meta-data description 12 has a separate meta-data parser 16. For example, a COBOL parser 18 would parse meta-data description 12 for a data structure written in the COBOL programming language, while a 3270 parser 20 would parse meta-data description 12 for a data structure written for a 3270 terminal emulation software interface. Other examples of meta-data parsers 16 are possible and are included within the scope of the present invention.

Each meta-data parser 16 divides meta-data description 12 into the plurality of logical units of data 14. For example, COBOL parser 18 divides meta-data description 12 for a data structure written in COBOL according to the known syntax of the COBOL programming language. Thus, each meta-data parser 16 must be specifically constructed for each meta-data description 12.

Each logical unit of data 14 is then passed to a meta-data translator 22. Meta-data translator 22 translates each logical unit of data 14 into an application component 24 during a translation process according to two sets of rules. The first set of rules is a set of internal translation rules 26, which is defined separately for each type of meta-data description 12. For example, set of internal translation rules 26 for a data structure written in COBOL are determined according to the known syntax of the COBOL programming language. These rules enable meta-data translator 22 to retrieve such information from logical unit of data 14 as a name for application component 24, a data type for application component 24, a display interface for application component 24, and so forth. Examples of data types for application component 24 include, but are not limited to, a scalar, a Boolean, an integer and a list. Thus, each set of internal translation rules 26 is preferably specifically constructed for each meta-data parser 16.

In addition, set of internal translation rules 26 enables meta-data translator 22 to create default values for definitions which cannot be determined according to the data available in logical unit of data 14. These default values are determined according to at least one assumption made in set of internal translation rules 26. For example, if a data for a particular variable is stored in the data of the enterprise resource, then this assumption could include a range of permissible values for that variable.

Preferably, at least one assumption would enable a mechanism for displaying the data in the GUI (graphical user interface) to be determined according to set of internal translation rules

26. For example, a group name and at least one variable field could be defined in COBOL code written to define the data structure of a particular enterprise resource. The default assumptions of set of internal translation rules 26 could optionally define the display mechanism for the associated data as a panel, with the group name as the title of the panel, and the value or values for the variable displayed in the panel. As another example, if the group name includes the word "occurs", such that the related data has more than one instance, then the data could optionally be displayed in a plurality of "tab cards" on the GUI.

The GUI is therefore preferably determined as an abstract GUI, composed of a plurality of abstract components, such as a check box for example. Each abstract component is more preferably selected from a group of such components, such that the properties of the abstract component are optionally and more preferably automatically known to processes which occur at run-time. Thus, preferably these rules enable assumptions concerning both valid data and mechanisms for displaying the data in the GUI to be defined during the translation process, such that the GUI is defined as a GUI abstraction with a plurality of abstract GUI components.

Set of internal translation rules 26 enables each application component 24 to be created by an application component factory 27 according to the attributes of a template for application component 24. Application component factory 27 is therefore able to use the information obtained according to set of internal translation rules 26 by meta-data translator 22 to enter the necessary information for each attribute in the template, thereby creating application component 24. More preferably, application component factory 27 is able to transform each application component 24 into an object, with associated data and methods for accessing the data. This preferred object-oriented architecture for each application component 24 also enables application component 24 to be placed within a hierarchical object-oriented structure for the new enterprise resource, described in greater detail below.

In addition, preferably the interface components (not shown), which are other associated components of the generated application, described in greater detail below with regard to Figure 2, are also generated by application component factory 27. These interface components are constructed from an interface, which provides the standard for defining each type of interface component. Application component factory 27 selects the correct interface components for each application component 24, as well as the correct attributes for each interface component, according to the translation process which was previously described.

The second set of rules for the translation process is a set of external translation rules 28. These external rules preferably include at least one rule defined by the programming user

of the system of the present invention, which either override one or more rules of set of internal translation rules 26, or alternatively are in addition to one or more rules of set of internal translation rules 26. Optionally and preferably set of external translation rules 28 includes at least one rule which is specific to the particular enterprise resource but which is not necessarily defined according to the syntax structure of the enterprise application. For example, such a rule could modify the length of the data or could define particular features of the mechanism for displaying the data on the GUI.

Meta-data parser 12 also determines a position 30 for each logical unit of data 14 within the hierarchical object oriented structure. Meta-data parser 12 preferably builds the tree dynamically, as each logical unit of data 14 as parsed, by determining the relationship between each parsed logical unit of data 14 and previously parsed logical units of data 14, for example as a parent, brother, or child of such a previously parsed logical unit of data 14.

Next, logical position 30 for each logical unit of data 14 is used to construct a hierarchical object-oriented structure 32 of application components 24 by a hierarchical structure constructor (not shown), which also includes the associated interface components for each application component 24. Hierarchical object-oriented structure 32 is preferably automatically adjusted to reduce repetition of data. Optionally and preferably, if a particular application component 24 recurs repeatedly throughout meta-data structure 12, preferably hierarchical object-oriented structure 32 is constructed to indicate the repeated occurrences of application component 24 without repeating all of the information about application component 24, for example by storing a pointer to application component 24.

Hierarchical object-oriented structure 32 is optionally displayed to the programming user by an editing environment (not shown), such that the programming user can then edit hierarchical object-oriented structure 32. For example, the programming user could choose to alter a position of one or more application components 24, to remove one or more application components 24 from hierarchical object-oriented structure 32, to adjust one or more associated interface components, or to add information which could not be automatically determined by meta-data translator 22 or application component factory 27. One example of such information would be specific values for one or more parameters for which default values were entered during the automatic translation process described previously. Such information is preferably entered by the programming user, since values for these parameter(s) are not available from the data in logical unit of data 14, or else cannot be determined automatically during the translation process. More preferably, hierarchical object-oriented structure 32 is displayed to the

programming user in the context of a development environment, such as a GUI (graphical user interface), which simplifies these changes and additions through “drag and drop” GUI gadgets and other aids to the programming user.

One advantage of system 10 of the present invention is that the rules of set of internal translation rules 26 is optionally simplified, in order to avoid complex rule-based systems which are difficult to operate. Rather, system 10 both enables the programming user to define additional, enterprise resource-specific rules in set of external translation rules 28, and to manually edit hierarchical object-oriented structure 32. Thus, system 10 both is simple to operate and yet still enables the programming user to adjust the constructed application as needed.

Hierarchical object-oriented structure 32 of application components 24 is then preferably transformed by a definition extraction factory 34 into a definitions file 36. Definition extraction factory 34 travels down through hierarchical object-oriented structure 32 and examines each application component 24 individually. Definition extraction factory 34 then extracts the necessary information from each application component 24 in order to construct definitions for each aspect of the new application for accessing the enterprise resource. Definition extraction factory 34 preferably also selects the correct interface components, with the correct attributes, for each application component 34. More preferably, definition extraction factory 34 includes a plurality of creators (not shown). Each particular creator is able to extract information from application component 24 in order to select the correct instance of an interface component, with the correct attributes, for a particular type of interface component.

Definitions file 36 is preferably a flat text file containing a list of these definitions for each application component 24 and for the associated interface components. However, definitions file 36 could also be a generic text file or a binary format file, for example. Definitions file 36 is then used to create the internal structure of application 38 at run-time, such that the definitions for each application component 24 are read, and are then used to create application 38.

Preferably, these definitions are parsed by a parsing engine (not shown) which is similar to application component factory 27. Since these definitions are determined according to the present invention, there is no requirement for translating the data structure or for assuming any default parameters. Instead, the parsing engine creates each application component 24, and then links each application component 24 to the interface components as described in greater

detail below. The overall structure of hierarchical object-oriented structure 32 is then created by the parsing engine to link all of these components together into application 38.

More preferably, parsing engine creates a framework for the components of application 38. This framework then optionally provides services for these components. Ultimately, the enterprise data is provided to a client (not shown), which then displays the data to the user. Therefore, one portion of the framework is most preferably an object which contains the client information, such as the type of communication language used by the client, as well as the type of GUI requested by the client. Examples of different types of communication languages include, but are not limited to, HTML (HyperText Mark-up Language); Java, which may have different sub-types such as Swing, AWT and so forth, XML (extensible mark-up language); and WML (Wireless Mark-up Language). Examples of different types of GUI include, but are not limited to, a Web browser, a Java client interface and no GUI. The latter type of GUI is preferred for applications, such as SQL (sequential query language) database applications by Oracle Ltd. or other such applications, which do not require a GUI. Thus, the client could optionally request not to receive a GUI for these types of applications, but otherwise preferably specifies a type of GUI for displaying the information.

Once the parsing engine receives this information, then the GUI is generated at run-time and/or the time of parsing (if this process occurs before the run-time for the client). Preferably, the GUI is generated by the parsing engine from the previously described GUI abstraction, with the abstract GUI components. The parsing engine also preferably requests a particular library of corresponding specific, implemented GUI components according to the type of communication protocol and the type of application. For example, one type of library might be relevant to a GUI designed for a Web browser and implemented in HTML, while a different type of library would be used for a Java interface GUI which is implemented in the Swing type of Java. Each library would contain information for implementing each type of abstract GUI component as an actual, concrete GUI component.

In addition, each library preferably includes a layout manager for determining the spatial relationship between GUI components. More preferably, each such component, whether concrete or abstract, is specified in relation to other GUI components, rather than with absolute sizes or other absolute properties, in order for the layout manager to operate more effectively. The layout manager is also optionally and more preferably embodied in a rules-based implementation, for determining properties of the GUI component such as size for example. Such a layout manager is available as part of the Java developers' toolkit (Sun Microsystems

Ltd.), but is also preferably implemented for other communication languages such as HTML for example.

Optionally and more preferably, the layout manager operates according to a selected layout policy, which again is most preferably abstracted to an abstract policy, similar to the GUI abstraction. Such an abstract policy enables the parsing engine, in combination with the layout manager, to more easily implement the actual GUI with the actual GUI layout policy, regardless of the client communication language type and/or the client application type. One advantage of such a set of abstractions is that the human software developer can more easily construct various types of GUI components and layout policies as a generalized GUI, rather than implementing the GUI specifically for each type of client communication language type and/or the client application type.

More preferably, at run-time adjustments are made to the structure of application 38 by the parsing engine. For example, the parsing engine may recognize a repetitive sequence within hierarchical object-oriented structure 32, with multiple sub-hierarchies associated with a particular application component 24. The parsing engine could then optionally reproduce portions of application 38 which correspond to the multiple sub-hierarchies, thereby more efficiently generating application 38. Preferably, the programming user can also optionally use this feature of system 10 in order to create part of hierarchical object-oriented structure 32 by selecting and then duplicating one or more application components 24.

Each interface component of application 38, apart from application component 24, is preferably defined according to an interface for that type of component. Preferably, the programming user is able to select interface components from a predefined set of interface components in order to construct application 38. Optionally and preferably, the programming user can create new interface components from the interface which is provided for each type of component, such that these new interface components are specific to the requirements of the particular enterprise resource for which the programming user is constructing application 38.

If the programming user wishes to alter one or more aspects of application 38 after definitions file 36 has been created, then definitions file 36 is read by definition extraction factory 34 to form all of the components, including application component 24 and the interface components of application 38 (not shown in Figure 1, see Figure 2), as objects. Each object can then be adjusted by the programming user, for example by altering one or more rules of set of external translation rules 28. The object is then processed again by application component factory 27 to form application component 24 and/or one or more of the interface components of

application 38.

Figure 2 shows a preferred implementation of the structure of application 38, which features components which can be divided into three levels of control of functions within application 38. In addition, the components are of two types. The first type of component is the interface components which were previously described. The second type of component provides the backbone for application 38, and includes application component 24 and extensions to application component 24.

The first of the three levels is a base level 40, which provides overall control for application 38. The backbone component of base level 40 is a base application component 42, which manages the interactions of the remaining components of application 38. Base application component 42 is the most extended from application component 24, and is preferably unique to application 38. Base application component 42 controls both the flow of data, including data import to and export from the enterprise data resource, and the display of the user interface to the application user.

An application control component 44 contains all of the rules and activities for importing data to, and exporting data from, the enterprise resource, which are preferably at least partially determined by the programming user. Preferably, the programming user enters a set of rules for how the enterprise resource can be accessed, for example for displaying certain types of data, and application control component 44 is then created at least partially according to these rules.

A user interface control component 46 provides control for the user interface, including for the functionality of the user interface and for the way in which the user interface is displayed to the user. Both application control component 44 and user interface control component 46 are controlled by base application component 42.

The next level of application 38 is a first node level 48. First node level 48 contains components for controlling data import to, and export from, the enterprise resource and for display of the retrieved data to the application user through the user interface. The backbone component for first node level 48 is an I/O component 50, which is also an extension of application component 24, although somewhat less extended in comparison to base application component 42. I/O component 50 controls the activities of an I/O control component 52 and of a user interface component 54, as well as passing data to be exported from the enterprise resource and displayed by the user interface according to user interface component 54, and data to be retrieved through the user interface and imported to the enterprise resource through I/O

control component **52**.

I/O control component **52** controls data input and output from the enterprise resource. Optionally and preferably, each application **38** has more than one I/O control component **52** for retrieving data from the enterprise resource according to different retrieval modes. For example, the data could be presented to the application user through a terminal emulation software interface or alternatively through a data structure interface written in COBOL.

Preferably, application **38** is coded as a group of Java-based data objects, since the Java programming language enables each I/O control component **52** to subscribe to particular types of events, and hence to “listen” only for activity which should be controlled by that particular I/O control component **52**.

User interface component **54** preferably includes instructions for displaying the retrieved data in a GUI container according to the attributes provided by I/O component **50**, such as whether the data is displayed in a table, through a set of “tab card” GUI display objects, or any other type of GUI display form. More preferably, the programming user is able to select a particular pattern for the display of the information in a GUI display object, such that user interface component **54** then displays the data according to the selected pattern. For example, if the data includes a number of repeated sequences of numbers of other data, preferably a pattern is selected which is suitable for repetitive data. Alternatively and preferably, such a pattern could be automatically selected during the previously described automatic conversion process. In addition, user interface component **54** preferably also includes at least one display property, such as color of text or symbols, size of GUI display object, background color, and other display properties for the data.

User interface component **54** optionally and preferably implements the user interface as a Java application, or alternatively as an applet. More preferably, user interface component **54** implements the user interface in a document mark-up language, such as HTML or XML for example, for display by a Web browser.

The next level of both definitions file **36** and application **38** is a second node level **56**. Second node level **56** contains objects which are specific to each application component **24**, including application component **24** itself. In addition, second node level **56** includes an import interface **58**, for importing data into the enterprise resource associated with application component **24**, and an export interface **60**, for retrieving data from the enterprise resource associated with application component **24**. Both import interface **58** and export interface **60** interact with I/O control component **52** of first node level **48**, which was previously described.

Optionally and preferably, a single application component **24** can have more than one associated import interface **58** and export interface **60**, for example in order to enable the enterprise resource to be accessed through multiple types of data access. However, preferably, each associated import interface **58** and export interface **60** is controlled by a single I/O control component **52**.

For the sake of symmetry, user interface component **54** is shown again in second node level **56**. It should be noted that user interface component **54** is present at both first node level **48** and second node level **56**, since the user interface is preferably provided through a single interface component as shown. However, optionally user interface component **54** could be divided into two interface components: a first such component for first node level **48**, which would provide a user interface for controlling data input and output; and a second such component for second node level **56**, which would actually display the exported data.

The operation of the components of first node level **48** and second node level **56** is preferably implemented as follows. I/O control component **52** controls the flow of data into and out from the enterprise resource, notifying the other components of application **38** which are subscribed to events related to particular types of data. For exporting data from the enterprise resource, I/O control component **52** receives the raw data from the resource. I/O control component **52** then analyzes the raw data, and notifies each export interface **60** which is subscribed to a portion of the raw data that such data is available. I/O control component **52** parses the data according to the data structure. For example, for data stored in a structure written in the COBOL programming language, the parsing instruction is based upon a fixed format which includes an offset and a length of the data field. Alternatively, the data may be separated by data separators, such as commas. Also alternatively, the data may be stored in the format "field name = field value". After parsing the data, I/O control component **52** then provides the appropriate portion of the raw data to each export interface **60**, which translates the raw data into the appropriate format for display through user interface component **54**.

Similarly, for importing data into the enterprise resource, user interface control component **46** informs the relevant I/O control component **52** to prepare to receive data for storage. I/O control component **52** then notifies all other subscribing components to provide data through export interface **60**. I/O control component **52** receives the data and prepares the data in the proper format for storage. Optionally and preferably, each I/O control component **52** sends such data to a different computer, through a different interface and based upon a different language. Thus, each I/O control component **52** could be used to control data input to,

and output from, a different enterprise resource, such that application 38 would provide an integrated solution to the management of these different enterprise resources.

A validation interface component 64 interacts with export interface 60, import interface 58, user interface component 54 and with a data structure interface component 66, in order to
5 validate the data associated with application component 24.

Data validation includes such functions as determining whether data to be retrieved from, or written to, a enterprise resource associated with application component 24 has a value or values which are valid for that type of data. In addition, data validation may include determining whether data can be modified, which in turn is optionally determined by the state
10 of application 38. For example, in one state, application 38 could be determined to be "read only", such that data can only be retrieved from the enterprise resource, but may not be written to the enterprise resource. In this state, import interface 58 should not be permitted to import new data and/or to change existing data. Each state preferably has a set of rules, for determining if data can be displayed, whether a particular object of application 32 is operative,
15 and so forth. Validation interface component 64 and data structure interface component 66 together provide low-level controls for maintaining the rules for each state of application 32.

For example, export interface 60 could retrieve data from the enterprise resource, which would be stored in data structure interface component 66. User interface component 54 would then retrieve the data stored in data structure interface component 66 for display to the
20 application user through the user interface. Thus, the mechanism for displaying the data is independent from the data itself, since export interface 60 translates the data into the correct format for display by user interface component 54 and since user interface component 54 does not need to interact directly with export interface 60.

Furthermore, one or more components of application 38 could be replaced with another
25 component of the same type, without altering the function of the remaining components. Thus, the structure of application 38 which is displayed in Figure 2 provides maximum flexibility for interacting with the enterprise resource.

It will be appreciated that the above descriptions are intended only to serve as examples,
30 and that many other embodiments are possible within the spirit and the scope of the present invention.

WHAT IS CLAIMED IS:

1. A system for automatic construction of an application for an enterprise resource for operation by a user, the system comprising:
 - (a) a meta-data description of the enterprise resource, said meta-data description featuring a plurality of logical units of data;
 - (b) a meta-data parser for parsing the enterprise resource into said plurality of logical units of data according to said meta-data description and for determining a relationship between said logical units of data;
 - (c) a meta-data translator for translating each logical unit of data into an application component, for associating said application component with at least one interface component for interfacing with the enterprise resource; and
 - (d) a definition extraction factory for creating the application from a plurality of said application components with said at least one interface component, at least according to said relationship between said plurality of logical units of data.
2. The system of claim 1, wherein said at least one interface component is a plurality of interface components, including at least one interface component for data flow to and from the enterprise resource and at least one interface component for providing a user interface to the user.
3. The system of claim 2, wherein the application further comprises:
 - (i) a first layer of interface components for providing control of the application, including at least one interface component for controlling said user interface and at least one interface component for controlling said data flow;
 - (ii) a second layer of interface components for providing said data input and said data output, including said at least one interface component for providing said user interface; and
 - (iii) a third layer for specifically interacting with the enterprise resource.
4. The system of claim 3, wherein said application component is contained in said third layer and said third layer further comprises:
 - (1) an export interface for retrieving data from the enterprise resource; and
 - (2) an import interface for storing data into the enterprise resource.

5. The system of claim 4, wherein each of said first layer and said second layer includes an extension of said application component for communicating with said at least one interface component related to said user interface and said at least one interface component related to said data flow.
6. The system of claim 5, wherein said at least one interface component is defined according to a standard interface and according to at least one user-defined function.
7. The system of claim 1, wherein said meta-data description, said meta-data parser and said meta-data translator are specific for a particular type of enterprise resource.
8. The system of claim 7, wherein said meta-data translator translates each logical unit of data according to at least one internal translation rule.
9. The system of claim 8, wherein said at least one internal translation rule is defined according to an attribute of said particular type of enterprise resource.
10. The system of claim 9, wherein said at least one internal translation rule further includes at least one default value for a parameter not described in said meta-data description.
11. The system of claim 10, wherein said meta-data translator also translates each logical unit of data according to at least one external translation rule defined by the user.
12. The system of claim 1, further comprising:
 - (e) a hierarchical structure constructor for arranging said plurality of application components into a hierarchical object-oriented structure according to said relationship between said logical units of data, such that said definition extraction factory also creates the application according to said hierarchical object-oriented structure.
13. The system of claim 12, further comprising:

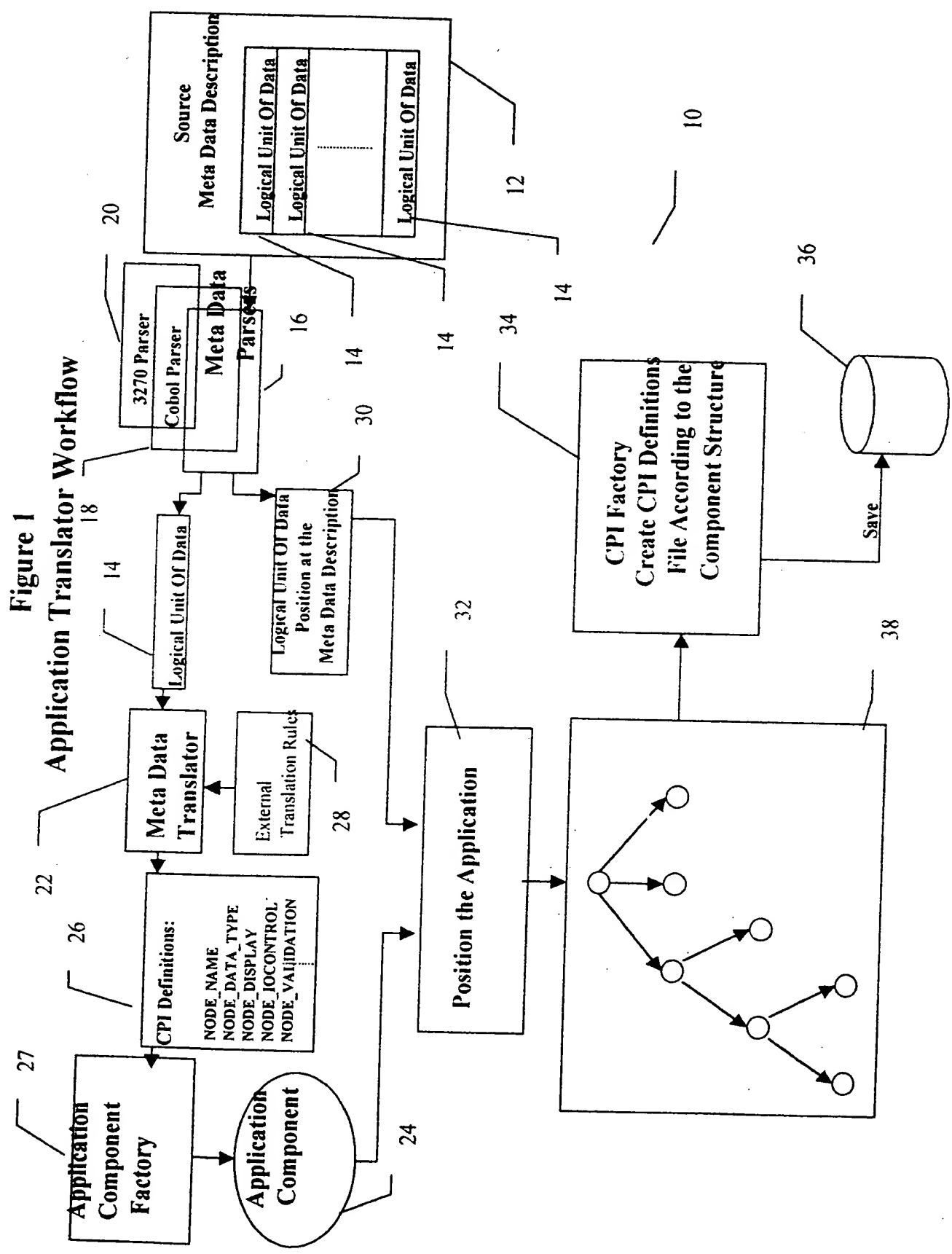
- (f) a definitions file for defining each of said plurality of application components, such that the application is stored as said definitions file, said definitions file being constructed by said definition extraction factory.
- 14. The system of claim 13, further comprising:
 - (g) an editing environment for editing said hierarchical object-oriented structure by the user, such that the user manually alters at least a portion of said hierarchical object oriented structure.
- 15. The system of claim 14, further comprising:
 - (h) a parsing engine for reading said definitions file and for operating the application according to said definitions file.
- 16. The system of claim 15, further comprising:
 - (i) a user interface of the application for interacting with the user; and
 - (j) a client for operating said user interface, such that said parsing engine constructs said user interface automatically according to at least one property of said client.
- 17. The system of claim 16, wherein said user interface is a GUI (graphical user interface), said GUI being constructed by said meta-data translator with at least one abstract GUI component, the system further comprising:
 - (i) at least one GUI library for containing a conversion of said at least one abstract GUI component to a concrete GUI component, such that said parsing engine also constructs said user interface automatically according to said GUI library.
- 18. The system of claim 17, wherein said at least one GUI library contains a plurality of concrete GUI components and a layout manager for determining a spatial relationship between said plurality of concrete GUI components.
- 19. A method for automatically constructing an application for an enterprise data resource for a user, the steps of the method being performed by a data processor, the method comprising the steps of:

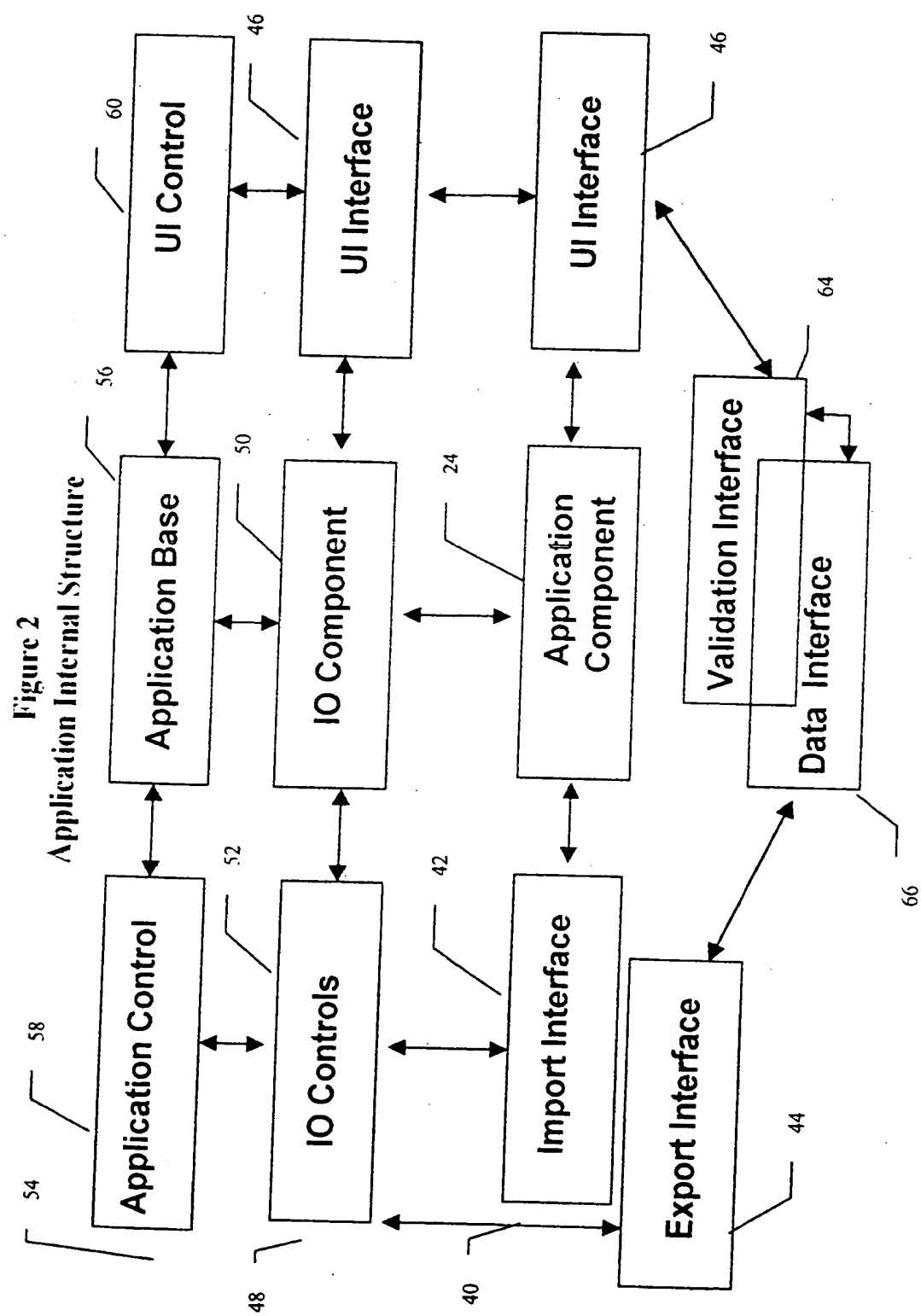
- (a) providing a meta-data description of the enterprise resource, said meta-data description including at least one attribute of the enterprise resource;
 - (b) dividing the enterprise resource into a plurality of logical data units according to said meta-data description;
 - (c) translating each of said plurality of logical data units into an application component according to at least one attribute of said meta-data description to form a plurality of application components;
 - (d) determining a relationship between said plurality of logical data units; and
 - (e) constructing the application according to said plurality of application components and said relationship between said plurality of logical data units.
20. The method of claim 19, wherein step (d) further comprises the step of:
- (i) constructing a hierarchical object-oriented structure for said plurality of application components according to said relationship between said plurality of logical data units.
21. The method of claim 20, wherein step (d) further comprises the steps of:
- (ii) providing an editing environment for displaying said hierarchical object-oriented structure to the user; and
 - (iii) altering at least a portion of said hierarchical object-oriented structure by the user.
22. The method of claim 21, wherein step (c) is performed according to at least one internal translation rule, said at least one internal translation rule being defined according to a type of the enterprise resource.
23. The method of claim 22, wherein step (c) is additionally performed according to the steps of:
- (i) defining at least one external translation rule by the user; and
 - (ii) translating each of said plurality of logical data units into an application component according to said at least one external translation rule.

24. The method of claim 23, further comprising the steps of:
- (f) extracting a plurality of definitions for describing each of said plurality of application components and said relationship; and
 - (g) storing the application by storing said plurality of definitions.
25. The method of claim 24, further comprising the step of:
- (h) operating the application according to said plurality of definitions.
26. The method of claim 25, further comprising the step of:
- (i) altering the application by changing at least one of said plurality of definitions.
27. The method of claim 22, wherein the application is operated by a client, and step (e) includes the steps of:
- (1) automatically constructing a user interface according to at least one property of said client; and
 - (2) displaying said user interface to the user for interacting with the user.
28. The method of claim 27, wherein step (1) further includes the steps of:
- (A) constructing an abstract user interface for displaying data from the enterprise resource and for interacting with the user; and
 - (B) constructing a concrete user interface according to said abstract user interface and according to said at least one property of said client.

ABSTRACT OF THE DISCLOSURE

A system and a method for automatic creation of a software application for accessing an enterprise data resource, preferably without actually altering the stored data. Rather, the system and method of the present invention create an interface which is preferably operable across computer platforms, and which enables the user to both write data to, and retrieve data from the enterprise resource. This is accomplished by first analyzing the enterprise resource to decompose it into a plurality of logical data units. Each data unit is then translated into an application component, which is preferably an object with associated methods and data. Each application component is then added to a hierarchical object-oriented structure, which is constructed according to the logical relationships between the units of data. The hierarchical object-oriented structure is optionally edited manually by the user. Finally, the hierarchical object-oriented structure is transformed into the application by an engine which uses a "factory" model, thereby efficiently coding those portions of the interface which are similar or identical for all enterprise resources. The final application is preferably a group of objects which provide such functions as a user interface and data input and output.





PCT

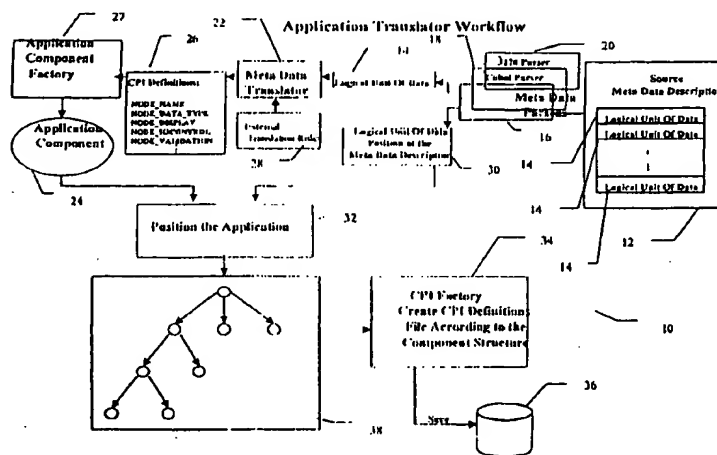
WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁷ : G06F 17/30		A1	(11) International Publication Number: WO 00/57300
			(43) International Publication Date: 28 September 2000 (28.09.00)
(21) International Application Number: PCT/IL00/00178		(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).	
(22) International Filing Date: 20 March 2000 (20.03.00)			
(30) Priority Data: 09/273,464 22 March 1999 (22.03.99) US			
(71) Applicant (for all designated States except US): INTERLINK COMPUTER COMMUNICATIONS LTD. [IL/IL]; Jabotinsky Street 5, Ramat-Gan 52520 (IL).			
(72) Inventor; and (75) Inventor/Applicant (for US only): HADAS, Ilan [IL/IL]; Hahadarim 127, Tzoran 42823 (IL).			
(74) Agent: PLINNER SA'AR, ADV.; Dizengoff Street 10, Tel Aviv 64281 (IL).		Published With international search report.	

(54) Title: AUTOMATIC INTERFACE GENERATION FOR AN ENTERPRISE RESOURCE



(57) Abstract

A system and method are presented for automatically accessing stored data via an enterprise data resource, without actually altering the stored data. Rather, the present invention creates an interface (46) that enables the user to write and retrieve data via the enterprise resource. First, the enterprise resource is analyzed and decomposed into logical data units (14). Next, each logical data unit (14) is translated into an application component (24), comprising an object associated with methods and data. Next, each application component (24) is added to a hierarchical object-oriented data structure (38) according to the relationships between logical data units (14). The data structure (38) may be manually edited by the user. Finally, the data structure is translated into an application by an engine using a "factory" model, thereby efficiently coding the portions of the interface that are similar or identical for all enterprise data resources. The final application is preferably a group of objects having a user interface, data input, and data output.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

AUTOMATIC INTERFACE GENERATION FOR AN ENTERPRISE RESOURCE

FIELD AND BACKGROUND OF THE INVENTION

5 The present invention relates to a system and method for automatically generating an application for interacting with an enterprise resource, and in particular, for such a system and method which constructs such an application without extensive manual intervention from a human programmer for an enterprise resource, such that the data associated with the enterprise resource is available through the application, for manipulation, export and import of data.

10 An enterprise resource is a structured set of data which is accessible through an enterprise application, for example by manipulating, exporting and importing data. The enterprise application is a software program which enables such access and manipulation of the data of the resource. One example of an enterprise resource is a database which is accessible through an interface written in the COBOL programming language. The term "enterprise"
15 refers to an existing resource and application within an organization, for example. Such enterprise resources may be maintained simply because constructing a completely new data structure and a new application interface is expensive and time-consuming, and may also result in significant "down time" until the new application is running smoothly and the enterprise application can be removed. Thus, organizations are hesitant to completely replace enterprise
20 applications and resources.

 A more useful system and method would enable the organization to integrate new or additional functionality to the enterprise application and resource, while still maintaining the enterprise application and resource, without requiring extensive programming by a human programmer and without requiring extensive testing before the new application and data
25 structure is ready for use. Such a system and method would be either semi-automatic, or even completely automatic, thereby reducing the possibility of human error as well as reducing the amount of human intervention required to prepare the new application and resource. Finally, such a system and method would produce an application which is widely usable across many different computer platforms, yet which is flexible and robust. Unfortunately, such a system
30 and method are not currently available.

 There is thus a need for, and it would be useful to have, a system and a method for automatic generation of a new resource data structure and of a software interface for that data structure, which would interface with an enterprise resource and an enterprise application, and

which would be flexible and robust, yet would not require substantial manual intervention by a human programmer for the construction of the new data structure and software interface, thereby integrating the enterprise application or applications into a single client interface.

5 **BRIEF DESCRIPTION OF THE DRAWINGS**

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, wherein:

FIG. 1 is a schematic block diagram of an illustrative system and work flow according
10 to the present invention; and

FIG. 2 is a schematic block diagram of an application created according to the present invention.

SUMMARY OF THE INVENTION

15 The present invention is of a system and a method for automatic creation of a software application for accessing an enterprise data resource, preferably without actually altering the stored data. Rather, the system and method of the present invention create an interface which is preferably operable across computer platforms, and which enables the user to both write data to, and retrieve data from the enterprise resource. This is accomplished by first analyzing the
20 enterprise resource to decompose it into a plurality of logical data units. Each data unit is then translated into an application component, which is preferably an object with associated methods and data. Each application component is then added to a hierarchical object-oriented structure, which is constructed according to the logical relationships between the units of data. The hierarchical object-oriented structure is optionally edited manually by the user. Finally, the
25 hierarchical object-oriented structure is transformed into the application by an engine which uses a "factory" model, thereby efficiently coding those portions of the interface which are similar or even identical for all enterprise resources. The final application is preferably a group of objects which provide such functions as a user interface and data input and output.

According to the present invention, there is provided a system for automatic
30 construction of an application for an enterprise resource for operation by a user, the system comprising: (a) a meta-data description of the enterprise resource, the meta-data description featuring a plurality of logical units of data; (b) a meta-data parser for parsing the enterprise resource into the plurality of logical units of data according to the meta-data description and for

determining a relationship between the logical units of data; (c) a meta-data translator for translating each logical unit of data into an application component, for associating the application component with at least one interface component for interfacing with the enterprise resource; and (d) a definition extraction factory for creating the application from a plurality of the application components with the at least one interface component, at least according to the relationship between the plurality of logical units of data.

According to another embodiment of the present invention, there is provided a method for automatically constructing an application for an enterprise data resource for a user, the steps of the method being performed by a data processor, the method comprising the steps of:

(a) providing a meta-data description of the enterprise resource, the meta-data description including at least one attribute of the enterprise resource; (b) dividing the enterprise resource into a plurality of logical data units according to the meta-data description; (c) translating each of the plurality of logical data units into an application component according to at least one attribute of the meta-data description to form a plurality of application components; (d) determining a relationship between the plurality of logical data units; and (e) constructing the application according to the plurality of application components and the relationship between the plurality of logical data units.

Hereinafter, the term "network" refers to a connection between any two computers which permits the transmission of data. Hereinafter, the term "computer" includes, but is not limited to, personal computers (PC) having an operating system such as DOS, Windows™, OS/2™ or Linux; Macintosh™ computers; computers having any type of Java virtual machine as the operating system; and graphical workstations such as the computers of Sun Microsystems™ and Silicon Graphics™, and other computers having some version of the UNIX operating system such as AIX™ or SOLARIS™ of Sun Microsystems™; or any other known and available operating system, including operating systems such as Windows CE™ for embedded systems, including cellular telephones, handheld computational devices and palmtop computational devices, and any other computational device which can be connected to a network. Hereinafter, the term "Windows™" includes but is not limited to Windows95™, Windows 3.x™ in which "x" is an integer such as "1", Windows NT™, Windows98™, Windows CE™ and any upgraded versions of these operating systems by Microsoft Corp. (USA).

Hereinafter, the term "Web browser" refers to any software program, which can display multimedia data such as text, graphics, or both, from Web pages on World Wide Web sites.

Hereinafter, the term "Web page" refers to any document available for download and display, including, but not limited to, documents written in a mark-up language such as HTML (Hyper-text Mark-up Language), VRML (Virtual Reality Modeling Language), dynamic HTML, XML (Extended Mark-up Language) or related computer languages thereof.

5 Hereinafter, the term "application user" refers to the individual operating the application which is constructed according to the present invention, while the term "programming user" refers to the human programmer who is constructing the application by interacting with the system of the present invention.

10 The present invention could be described as a series of steps implemented by a data processor, such that the present invention could be implemented as hardware, software or firmware, or a combination thereof. For the present invention, a software application could be written in substantially suitable programming language, which could easily be selected by one of ordinary skill in the art. The programming language chosen should be compatible with the computer by which the software application is executed. Examples of suitable programming
15 languages include, but are not limited to, C, C++ and Java.

DETAILED DESCRIPTION OF THE INVENTION

20 The present invention is of a system and a method for automatic creation of a software application for accessing an enterprise data resource, preferably without actually altering the stored data. Rather, the system and method of the present invention create an application which is preferably operable across computer platforms, and which enables the application user to both write data to, and retrieve data from the enterprise resource. This is accomplished by first analyzing the enterprise resource to decompose it into a plurality of logical data units. Each data unit is then translated into an application component, which is preferably an object with
25 associated methods and data. Each application component is then added to a hierarchical structure which is preferably object-oriented. This hierarchical object-oriented structure is constructed according to the logical relationships between the units of data. By "object-oriented", it is meant that the hierarchical object-oriented structure features a plurality of objects, with data and methods, which are linked according to the relationships between
30 these objects. The hierarchical object-oriented structure is optionally edited manually by the programming user. Finally, the hierarchical object-oriented structure is transformed into the application by an engine which uses a "factory" model, thereby efficiently coding those portions of the interface which are similar or even identical for all enterprise resources. The

final application is preferably constructed as a group of component objects which provide such functions as a user interface and data input and output.

The final application is preferably adjusted to the requirements of a requesting client at run-time, particularly with regard to the GUI. For example, if the client which is requesting
5 access to the enterprise resource data is a thin client operating a Web browser through HTML, then preferably the GUI is implemented as a Web page with HTML forms. More preferably, the GUI is first constructed as a GUI abstraction, thereby enabling different specific, concrete GUI implementations to be constructed "on the fly" as described in greater detail below.

The term "enterprise resource" includes such resources as a database or other data
10 storage structure written in the programming language COBOL, for example, or a terminal emulator interface which enables the application user to access data through a particular type of terminal. Other examples of such enterprise resources are also possible within the scope of the present invention. However, for the purposes of discussion only, and without intending to be limiting in any way, the following description centers upon two types of enterprise resources: a
15 data storage structure written in COBOL, and a terminal emulation software interface.

The principles and operation of a method and system according to the present invention may be better understood with reference to the drawings and the accompanying description, it being understood that these drawings are given for illustrative purposes only and are not meant to be limiting.

Referring now to the drawings, Figure 1 is a schematic block diagram of the workflow
20 through the software modules in a system 10 for automatic generation of a new software application for an enterprise resource, preferably as a hierarchical object-oriented structure. A system 10 initially receives a meta-data description 12 of the data structure of the enterprise resource. This meta-data description divides the data of the enterprise resource into a plurality
25 of logical data units 14. For example, for a data structure written in the COBOL programming language, each logical data unit 14 would be determined according to the syntax of the COBOL programming language. This syntax divides the data into different types, each of which is stored in a particular field. Thus, for a data structure written in COBOL, one example of a logical data unit 14 would be a particular type of data field.

30 As another example, for a terminal emulation software program such as a terminal emulation program for the 3270 terminal type, meta-data description 12 would include information obtained from trapping a plurality of screen displays for that terminal. Again, each screen display could be divided into data entry fields, for example, such that each data entry

field could be an example of logical data unit 14.

Meta-data description 12 for a particular enterprise resource is then received by a meta-data parser 16. Preferably, each meta-data description 12 has a separate meta-data parser 16. For example, a COBOL parser 18 would parse meta-data description 12 for a data structure written in the COBOL programming language, while a 3270 parser 20 would parse meta-data description 12 for a data structure written for a 3270 terminal emulation software interface. Other examples of meta-data parsers 16 are possible and are included within the scope of the present invention.

Each meta-data parser 16 divides meta-data description 12 into the plurality of logical units of data 14. For example, COBOL parser 18 divides meta-data description 12 for a data structure written in COBOL according to the known syntax of the COBOL programming language. Thus, each meta-data parser 16 must be specifically constructed for each meta-data description 12.

Each logical unit of data 14 is then passed to a meta-data translator 22. Meta-data translator 22 translates each logical unit of data 14 into an application component 24 during a translation process according to two sets of rules. The first set of rules is a set of internal translation rules 26, which is defined separately for each type of meta-data description 12. For example, set of internal translation rules 26 for a data structure written in COBOL are determined according to the known syntax of the COBOL programming language. These rules enable meta-data translator 22 to retrieve such information from logical unit of data 14 as a name for application component 24, a data type for application component 24, a display interface for application component 24, and so forth. Examples of data types for application component 24 include, but are not limited to, a scalar, a Boolean, an integer and a list. Thus, each set of internal translation rules 26 is preferably specifically constructed for each meta-data parser 16.

In addition, set of internal translation rules 26 enables meta-data translator 22 to create default values for definitions which cannot be determined according to the data available in logical unit of data 14. These default values are determined according to at least one assumption made in set of internal translation rules 26. For example, if a data for a particular variable is stored in the data of the enterprise resource, then this assumption could include a range of permissible values for that variable.

Preferably, at least one assumption would enable a mechanism for displaying the data in the GUI (graphical user interface) to be determined according to set of internal translation rules

26. For example, a group name and at least one variable field could be defined in COBOL code written to define the data structure of a particular enterprise resource. The default assumptions of set of internal translation rules 26 could optionally define the display mechanism for the associated data as a panel, with the group name as the title of the panel, and the value or values for the variable displayed in the panel. As another example, if the group name includes the word "occurs", such that the related data has more than one instance, then the data could optionally be displayed in a plurality of "tab cards" on the GUI.

The GUI is therefore preferably determined as an abstract GUI, composed of a plurality of abstract components, such as a check box for example. Each abstract component is more preferably selected from a group of such components, such that the properties of the abstract component are optionally and more preferably automatically known to processes which occur at run-time. Thus, preferably these rules enable assumptions concerning both valid data and mechanisms for displaying the data in the GUI to be defined during the translation process, such that the GUI is defined as a GUI abstraction with a plurality of abstract GUI components.

Set of internal translation rules 26 enables each application component 24 to be created by an application component factory 27 according to the attributes of a template for application component 24. Application component factory 27 is therefore able to use the information obtained according to set of internal translation rules 26 by meta-data translator 22 to enter the necessary information for each attribute in the template, thereby creating application component 24. More preferably, application component factory 27 is able to transform each application component 24 into an object, with associated data and methods for accessing the data. This preferred object-oriented architecture for each application component 24 also enables application component 24 to be placed within a hierarchical object-oriented structure for the new enterprise resource, described in greater detail below.

In addition, preferably the interface components (not shown), which are other associated components of the generated application, described in greater detail below with regard to Figure 2, are also generated by application component factory 27. These interface components are constructed from an interface, which provides the standard for defining each type of interface component. Application component factory 27 selects the correct interface components for each application component 24, as well as the correct attributes for each interface component, according to the translation process which was previously described.

The second set of rules for the translation process is a set of external translation rules 28. These external rules preferably include at least one rule defined by the programming user

of the system of the present invention, which either override one or more rules of set of internal translation rules 26, or alternatively are in addition to one or more rules of set of internal translation rules 26. Optionally and preferably set of external translation rules 28 includes at least one rule which is specific to the particular enterprise resource but which is not necessarily defined according to the syntax structure of the enterprise application. For example, such a rule could modify the length of the data or could define particular features of the mechanism for displaying the data on the GUI.

Meta-data parser 12 also determines a position 30 for each logical unit of data 14 within the hierarchical object oriented structure. Meta-data parser 12 preferably builds the tree dynamically, as each logical unit of data 14 as parsed, by determining the relationship between each parsed logical unit of data 14 and previously parsed logical units of data 14, for example as a parent, brother, or child of such a previously parsed logical unit of data 14.

Next, logical position 30 for each logical unit of data 14 is used to construct a hierarchical object-oriented structure 32 of application components 24 by a hierarchical structure constructor (not shown), which also includes the associated interface components for each application component 24. Hierarchical object-oriented structure 32 is preferably automatically adjusted to reduce repetition of data. Optionally and preferably, if a particular application component 24 recurs repeatedly throughout meta-data structure 12, preferably hierarchical object-oriented structure 32 is constructed to indicate the repeated occurrences of application component 24 without repeating all of the information about application component 24, for example by storing a pointer to application component 24.

Hierarchical object-oriented structure 32 is optionally displayed to the programming user by an editing environment (not shown), such that the programming user can then edit hierarchical object-oriented structure 32. For example, the programming user could choose to alter a position of one or more application components 24, to remove one or more application components 24 from hierarchical object-oriented structure 32, to adjust one or more associated interface components, or to add information which could not be automatically determined by meta-data translator 22 or application component factory 27. One example of such information would be specific values for one or more parameters for which default values were entered during the automatic translation process described previously. Such information is preferably entered by the programming user, since values for these parameter(s) are not available from the data in logical unit of data 14, or else cannot be determined automatically during the translation process. More preferably, hierarchical object-oriented structure 32 is displayed to the

programming user in the context of a development environment, such as a GUI (graphical user interface), which simplifies these changes and additions through “drag and drop” GUI gadgets and other aids to the programming user.

One advantage of system 10 of the present invention is that the rules of set of internal translation rules 26 is optionally simplified, in order to avoid complex rule-based systems which are difficult to operate. Rather, system 10 both enables the programming user to define additional, enterprise resource-specific rules in set of external translation rules 28, and to manually edit hierarchical object-oriented structure 32. Thus, system 10 both is simple to operate and yet still enables the programming user to adjust the constructed application as needed.

Hierarchical object-oriented structure 32 of application components 24 is then preferably transformed by a definition extraction factory 34 into a definitions file 36. Definition extraction factory 34 travels down through hierarchical object-oriented structure 32 and examines each application component 24 individually. Definition extraction factory 34 then extracts the necessary information from each application component 24 in order to construct definitions for each aspect of the new application for accessing the enterprise resource. Definition extraction factory 34 preferably also selects the correct interface components, with the correct attributes, for each application component 34. More preferably, definition extraction factory 34 includes a plurality of creators (not shown). Each particular creator is able to extract information from application component 24 in order to select the correct instance of an interface component, with the correct attributes, for a particular type of interface component.

Definitions file 36 is preferably a flat text file containing a list of these definitions for each application component 24 and for the associated interface components. However, definitions file 36 could also be a generic text file or a binary format file, for example. Definitions file 36 is then used to create the internal structure of application 38 at run-time, such that the definitions for each application component 24 are read, and are then used to create application 38.

Preferably, these definitions are parsed by a parsing engine (not shown) which is similar to application component factory 27. Since these definitions are determined according to the present invention, there is no requirement for translating the data structure or for assuming any default parameters. Instead, the parsing engine creates each application component 24, and then links each application component 24 to the interface components as described in greater

detail below. The overall structure of hierarchical object-oriented structure 32 is then created by the parsing engine to link all of these components together into application 38.

More preferably, parsing engine creates a framework for the components of application 38. This framework then optionally provides services for these components. Ultimately, the enterprise data is provided to a client (not shown), which then displays the data to the user. Therefore, one portion of the framework is most preferably an object which contains the client information, such as the type of communication language used by the client, as well as the type of GUI requested by the client. Examples of different types of communication languages include, but are not limited to, HTML (HyperText Mark-up Language); Java, which may have different sub-types such as Swing, AWT and so forth, XML (extensible mark-up language); and WML (Wireless Mark-up Language). Examples of different types of GUI include, but are not limited to, a Web browser, a Java client interface and no GUI. The latter type of GUI is preferred for applications, such as SQL (sequential query language) database applications by Oracle Ltd. or other such applications, which do not require a GUI. Thus, the client could optionally request not to receive a GUI for these types of applications, but otherwise preferably specifies a type of GUI for displaying the information.

Once the parsing engine receives this information, then the GUI is generated at run-time and/or the time of parsing (if this process occurs before the run-time for the client). Preferably, the GUI is generated by the parsing engine from the previously described GUI abstraction, with the abstract GUI components. The parsing engine also preferably requests a particular library of corresponding specific, implemented GUI components according to the type of communication protocol and the type of application. For example, one type of library might be relevant to a GUI designed for a Web browser and implemented in HTML, while a different type of library would be used for a Java interface GUI which is implemented in the Swing type of Java. Each library would contain information for implementing each type of abstract GUI component as an actual, concrete GUI component.

In addition, each library preferably includes a layout manager for determining the spatial relationship between GUI components. More preferably, each such component, whether concrete or abstract, is specified in relation to other GUI components, rather than with absolute sizes or other absolute properties, in order for the layout manager to operate more effectively. The layout manager is also optionally and more preferably embodied in a rules-based implementation, for determining properties of the GUI component such as size for example. Such a layout manager is available as part of the Java developers' toolkit (Sun Microsystems

Ltd.), but is also preferably implemented for other communication languages such as HTML for example.

Optionally and more preferably, the layout manager operates according to a selected layout policy, which again is most preferably abstracted to an abstract policy, similar to the GUI abstraction. Such an abstract policy enables the parsing engine, in combination with the layout manager, to more easily implement the actual GUI with the actual GUI layout policy, regardless of the client communication language type and/or the client application type. One advantage of such a set of abstractions is that the human software developer can more easily construct various types of GUI components and layout policies as a generalized GUI, rather than implementing the GUI specifically for each type of client communication language type and/or the client application type.

More preferably, at run-time adjustments are made to the structure of application 38 by the parsing engine. For example, the parsing engine may recognize a repetitive sequence within hierarchical object-oriented structure 32, with multiple sub-hierarchies associated with a particular application component 24. The parsing engine could then optionally reproduce portions of application 38 which correspond to the multiple sub-hierarchies, thereby more efficiently generating application 38. Preferably, the programming user can also optionally use this feature of system 10 in order to create part of hierarchical object-oriented structure 32 by selecting and then duplicating one or more application components 24.

Each interface component of application 38, apart from application component 24, is preferably defined according to an interface for that type of component. Preferably, the programming user is able to select interface components from a predefined set of interface components in order to construct application 38. Optionally and preferably, the programming user can create new interface components from the interface which is provided for each type of component, such that these new interface components are specific to the requirements of the particular enterprise resource for which the programming user is constructing application 38.

If the programming user wishes to alter one or more aspects of application 38 after definitions file 36 has been created, then definitions file 36 is read by definition extraction factory 34 to form all of the components, including application component 24 and the interface components of application 38 (not shown in Figure 1, see Figure 2), as objects. Each object can then be adjusted by the programming user, for example by altering one or more rules of set of external translation rules 28. The object is then processed again by application component factory 27 to form application component 24 and/or one or more of the interface components of

application 38.

Figure 2 shows a preferred implementation of the structure of application 38, which features components which can be divided into three levels of control of functions within application 38. In addition, the components are of two types. The first type of component is the interface components which were previously described. The second type of component provides the backbone for application 38, and includes application component 24 and extensions to application component 24.

The first of the three levels is a base level 40, which provides overall control for application 38. The backbone component of base level 40 is a base application component 42, which manages the interactions of the remaining components of application 38. Base application component 42 is the most extended from application component 24, and is preferably unique to application 38. Base application component 42 controls both the flow of data, including data import to and export from the enterprise data resource, and the display of the user interface to the application user.

An application control component 44 contains all of the rules and activities for importing data to, and exporting data from, the enterprise resource, which are preferably at least partially determined by the programming user. Preferably, the programming user enters a set of rules for how the enterprise resource can be accessed, for example for displaying certain types of data, and application control component 44 is then created at least partially according to these rules.

A user interface control component 46 provides control for the user interface, including for the functionality of the user interface and for the way in which the user interface is displayed to the user. Both application control component 44 and user interface control component 46 are controlled by base application component 42.

The next level of application 38 is a first node level 48. First node level 48 contains components for controlling data import to, and export from, the enterprise resource and for display of the retrieved data to the application user through the user interface. The backbone component for first node level 48 is an I/O component 50, which is also an extension of application component 24, although somewhat less extended in comparison to base application component 42. I/O component 50 controls the activities of an I/O control component 52 and of a user interface component 54, as well as passing data to be exported from the enterprise resource and displayed by the user interface according to user interface component 54, and data to be retrieved through the user interface and imported to the enterprise resource through I/O

control component 52.

I/O control component 52 controls data input and output from the enterprise resource. Optionally and preferably, each application 38 has more than one I/O control component 52 for retrieving data from the enterprise resource according to different retrieval modes. For example, the data could be presented to the application user through a terminal emulation software interface or alternatively through a data structure interface written in COBOL.

Preferably, application 38 is coded as a group of Java-based data objects, since the Java programming language enables each I/O control component 52 to subscribe to particular types of events, and hence to “listen” only for activity which should be controlled by that particular I/O control component 52.

User interface component 54 preferably includes instructions for displaying the retrieved data in a GUI container according to the attributes provided by I/O component 50, such as whether the data is displayed in a table, through a set of “tab card” GUI display objects, or any other type of GUI display form. More preferably, the programming user is able to select a particular pattern for the display of the information in a GUI display object, such that user interface component 54 then displays the data according to the selected pattern. For example, if the data includes a number of repeated sequences of numbers of other data, preferably a pattern is selected which is suitable for repetitive data. Alternatively and preferably, such a pattern could be automatically selected during the previously described automatic conversion process. In addition, user interface component 54 preferably also includes at least one display property, such as color of text or symbols, size of GUI display object, background color, and other display properties for the data.

User interface component 54 optionally and preferably implements the user interface as a Java application, or alternatively as an applet. More preferably, user interface component 54 implements the user interface in a document mark-up language, such as HTML or XML for example, for display by a Web browser.

The next level of both definitions file 36 and application 38 is a second node level 56. Second node level 56 contains objects which are specific to each application component 24, including application component 24 itself. In addition, second node level 56 includes an import interface 58, for importing data into the enterprise resource associated with application component 24, and an export interface 60, for retrieving data from the enterprise resource associated with application component 24. Both import interface 58 and export interface 60 interact with I/O control component 52 of first node level 48, which was previously described.

Optionally and preferably, a single application component 24 can have more than one associated import interface 58 and export interface 60, for example in order to enable the enterprise resource to be accessed through multiple types of data access. However, preferably, each associated import interface 58 and export interface 60 is controlled by a single I/O control component 52.

For the sake of symmetry, user interface component 54 is shown again in second node level 56. It should be noted that user interface component 54 is present at both first node level 48 and second node level 56, since the user interface is preferably provided through a single interface component as shown. However, optionally user interface component 54 could be divided into two interface components: a first such component for first node level 48, which would provide a user interface for controlling data input and output; and a second such component for second node level 56, which would actually display the exported data.

The operation of the components of first node level 48 and second node level 56 is preferably implemented as follows. I/O control component 52 controls the flow of data into and out from the enterprise resource, notifying the other components of application 38 which are subscribed to events related to particular types of data. For exporting data from the enterprise resource, I/O control component 52 receives the raw data from the resource. I/O control component 52 then analyzes the raw data, and notifies each export interface 60 which is subscribed to a portion of the raw data that such data is available. I/O control component 52 parses the data according to the data structure. For example, for data stored in a structure written in the COBOL programming language, the parsing instruction is based upon a fixed format which includes an offset and a length of the data field. Alternatively, the data may be separated by data separators, such as commas. Also alternatively, the data may be stored in the format "field name = field value". After parsing the data, I/O control component 52 then provides the appropriate portion of the raw data to each export interface 60, which translates the raw data into the appropriate format for display through user interface component 54.

Similarly, for importing data into the enterprise resource, user interface control component 46 informs the relevant I/O control component 52 to prepare to receive data for storage. I/O control component 52 then notifies all other subscribing components to provide data through export interface 60. I/O control component 52 receives the data and prepares the data in the proper format for storage. Optionally and preferably, each I/O control component 52 sends such data to a different computer, through a different interface and based upon a different language. Thus, each I/O control component 52 could be used to control data input to,

and output from, a different enterprise resource, such that application 38 would provide an integrated solution to the management of these different enterprise resources.

A validation interface component 64 interacts with export interface 60, import interface 58, user interface component 54 and with a data structure interface component 66, in order to
5 validate the data associated with application component 24.

Data validation includes such functions as determining whether data to be retrieved from, or written to, a enterprise resource associated with application component 24 has a value or values which are valid for that type of data. In addition, data validation may include determining whether data can be modified, which in turn is optionally determined by the state
10 of application 38. For example, in one state, application 38 could be determined to be "read only", such that data can only be retrieved from the enterprise resource, but may not be written to the enterprise resource. In this state, import interface 58 should not be permitted to import new data and/or to change existing data. Each state preferably has a set of rules, for determining if data can be displayed, whether a particular object of application 32 is operative,
15 and so forth. Validation interface component 64 and data structure interface component 66 together provide low-level controls for maintaining the rules for each state of application 32.

For example, export interface 60 could retrieve data from the enterprise resource, which would be stored in data structure interface component 66. User interface component 54 would then retrieve the data stored in data structure interface component 66 for display to the
20 application user through the user interface. Thus, the mechanism for displaying the data is independent from the data itself, since export interface 60 translates the data into the correct format for display by user interface component 54 and since user interface component 54 does not need to interact directly with export interface 60.

Furthermore, one or more components of application 38 could be replaced with another
25 component of the same type, without altering the function of the remaining components. Thus, the structure of application 38 which is displayed in Figure 2 provides maximum flexibility for interacting with the enterprise resource.

It will be appreciated that the above descriptions are intended only to serve as examples,
30 and that many other embodiments are possible within the spirit and the scope of the present invention.

WHAT IS CLAIMED IS:

1. A system for automatic construction of an application for an enterprise resource for operation by a user, the system comprising:
 - (a) a meta-data description of the enterprise resource, said meta-data description featuring a plurality of logical units of data;
 - (b) a meta-data parser for parsing the enterprise resource into said plurality of logical units of data according to said meta-data description and for determining a relationship between said logical units of data;
 - (c) a meta-data translator for translating each logical unit of data into an application component, for associating said application component with at least one interface component for interfacing with the enterprise resource; and
 - (d) a definition extraction factory for creating the application from a plurality of said application components with said at least one interface component, at least according to said relationship between said plurality of logical units of data.
2. The system of claim 1, wherein said at least one interface component is a plurality of interface components, including at least one interface component for data flow to and from the enterprise resource and at least one interface component for providing a user interface to the user.
3. The system of claim 2, wherein the application further comprises:
 - (i) a first layer of interface components for providing control of the application, including at least one interface component for controlling said user interface and at least one interface component for controlling said data flow;
 - (ii) a second layer of interface components for providing said data input and said data output, including said at least one interface component for providing said user interface; and
 - (iii) a third layer for specifically interacting with the enterprise resource.
4. The system of claim 3, wherein said application component is contained in said third layer and said third layer further comprises:
 - (1) an export interface for retrieving data from the enterprise resource; and
 - (2) an import interface for storing data into the enterprise resource.

5. The system of claim 4, wherein each of said first layer and said second layer includes an extension of said application component for communicating with said at least one interface component related to said user interface and said at least one interface component related to said data flow.

6. The system of claim 5, wherein said at least one interface component is defined according to a standard interface and according to at least one user-defined function.

7. The system of claim 1, wherein said meta-data description, said meta-data parser and said meta-data translator are specific for a particular type of enterprise resource.

8. The system of claim 7, wherein said meta-data translator translates each logical unit of data according to at least one internal translation rule.

9. The system of claim 8, wherein said at least one internal translation rule is defined according to an attribute of said particular type of enterprise resource.

10. The system of claim 9, wherein said at least one internal translation rule further includes at least one default value for a parameter not described in said meta-data description.

11. The system of claim 10, wherein said meta-data translator also translates each logical unit of data according to at least one external translation rule defined by the user.

12. The system of claim 1, further comprising:

(e) a hierarchical structure constructor for arranging said plurality of application components into a hierarchical object-oriented structure according to said relationship between said logical units of data, such that said definition extraction factory also creates the application according to said hierarchical object-oriented structure.

13. The system of claim 12, further comprising:

- (f) a definitions file for defining each of said plurality of application components, such that the application is stored as said definitions file, said definitions file being constructed by said definition extraction factory.

14. The system of claim 13, further comprising:

- (g) an editing environment for editing said hierarchical object-oriented structure by the user, such that the user manually alters at least a portion of said hierarchical object oriented structure.

15. The system of claim 14, further comprising:

- (h) a parsing engine for reading said definitions file and for operating the application according to said definitions file.

16. The system of claim 15, further comprising:

- (i) a user interface of the application for interacting with the user; and
- (j) a client for operating said user interface, such that said parsing engine constructs said user interface automatically according to at least one property of said client.

17. The system of claim 16, wherein said user interface is a GUI (graphical user interface), said GUI being constructed by said meta-data translator with at least one abstract GUI component, the system further comprising:

- (i) at least one GUI library for containing a conversion of said at least one abstract GUI component to a concrete GUI component, such that said parsing engine also constructs said user interface automatically according to said GUI library.

18. The system of claim 17, wherein said at least one GUI library contains a plurality of concrete GUI components and a layout manager for determining a spatial relationship between said plurality of concrete GUI components.

19. A method for automatically constructing an application for an enterprise data resource for a user, the steps of the method being performed by a data processor, the method comprising the steps of:

- (a) providing a meta-data description of the enterprise resource, said meta-data description including at least one attribute of the enterprise resource;
- (b) dividing the enterprise resource into a plurality of logical data units according to said meta-data description;
- (c) translating each of said plurality of logical data units into an application component according to at least one attribute of said meta-data description to form a plurality of application components;
- (d) determining a relationship between said plurality of logical data units; and
- (e) constructing the application according to said plurality of application components and said relationship between said plurality of logical data units.

20. The method of claim 19, wherein step (d) further comprises the step of:

- (i) constructing a hierarchical object-oriented structure for said plurality of application components according to said relationship between said plurality of logical data units.

21. The method of claim 20, wherein step (d) further comprises the steps of:

- (ii) providing an editing environment for displaying said hierarchical object-oriented structure to the user; and
- (iii) altering at least a portion of said hierarchical object-oriented structure by the user.

22. The method of claim 21, wherein step (c) is performed according to at least one internal translation rule, said at least one internal translation rule being defined according to a type of the enterprise resource.

23. The method of claim 22, wherein step (c) is additionally performed according to the steps of:

- (i) defining at least one external translation rule by the user; and
- (ii) translating each of said plurality of logical data units into an application component according to said at least one external translation rule.

24. The method of claim 23, further comprising the steps of:
 - (f) extracting a plurality of definitions for describing each of said plurality of application components and said relationship; and
 - (g) storing the application by storing said plurality of definitions.
25. The method of claim 24, further comprising the step of:
 - (h) operating the application according to said plurality of definitions.
26. The method of claim 25, further comprising the step of:
 - (i) altering the application by changing at least one of said plurality of definitions.
27. The method of claim 22, wherein the application is operated by a client, and step (e) includes the steps of:
 - (1) automatically constructing a user interface according to at least one property of said client; and
 - (2) displaying said user interface to the user for interacting with the user.
28. The method of claim 27, wherein step (1) further includes the steps of:
 - (A) constructing an abstract user interface for displaying data from the enterprise resource and for interacting with the user; and
 - (B) constructing a concrete user interface according to said abstract user interface and according to said at least one property of said client.

